Seymour CRAY's Reference Manual for the 7600

CONTROL DATA®
7600 COMPUTER SYSTEM

Information presented in this edition is preliminary and subject to change. Any corrections necessitated by design changes and/or product improvement will be handled by standard manual revision procedures. Errors and suggestions should be communicated to Development Division, Technical Publications Dept.

PRELIMINARY
REFERENCE MANUAL

Most of the information in this manual was available early in 1967
The 7600 prototype checkout began around mid 1967
The first 7600 shipped 2/1/69

This document, along with expanded logic diagrams & wire lists, was used by the engineers when checking out the prototype, software people, the CE's during the early days of the 7600.

This manual is still used today by individuals in CDC's 7600 software support department.

FOREWORD

The ideas and designs set forth in this Control Data 7600 Preliminary Reference Manual are the property of Control Data Corporation and are not to be disseminated, distributed, or conveyed to third persons without the express written permission of the Control Data Corporation Patent Department.

TABLE OF CONTENTS

PART : SYSTEM DESCRIPTION

PART 2: CPU DESCRIPTION

PART 3: CPU INSTRUCTIONS

PART 4: PPU DESCRIPTION

PART 5: PPU INSTRUCTIONS

FIGURES

1 SYSTEM DESCRIPTION

PART I: SYSTEM DESCRIPTION

Introduction

The 7600 system is the result of a development program to provide computing capacity substantially beyond that of the 6600 systems. Central processor computation is expected to average four times as fast as corresponding computation in the 6600 system. The 7600 system is intended to be machine code upward compatible with the 6400/6600 systems in the area of central processor routines. It is not compatible on the machine code level in the area of system programs or input-output drivers. The 7600 system input-output provisions have been generalized and greatly expanded over those provided in the 6400/6600 systems. Input-output data rates are not expected to average substantially higher on a per channel basis than the rates in 6600 systems. A much larger volume of input-output data is handled in the 7600 system by a much larger number of input-output channels.

The 7600 system contains a central processing unit (CPU) and a number of peripheral processing units (PPU). Some of the PPU are physically located with the CPU and others may be remotely located. The PPU communicate with the CPU over high speed data links with the data buffered at the CPU end of the data link. The CPU is interrupted by the PPU once per data record, or on prescribed quantities of buffer data for long records. The PPU provide a communication and message switching function between the CPU and individual peripheral equipment controllers. Each PPU has a number of high speed data links to individual peripheral equipment controllers in addition to the data link to the CPU. The PPU time shares the data link to the CPU among the peripheral equipment controllers on a record by record basis.

The 7600 system is designed to accommodate multiple operating stations Each operating station is organized around a PPU which communicates directly with the CPU over its associated data link. New peripheral equipment configurations are being developed which will operate from programable equipment controllers and are specifically intended for this application. These controllers will be able to communicate with the PPU on a record by record basis at higher rates than the existing 6400/6600 equipment.

7600 system parameters

CPU computation section
- 60 bit internal word
- binary computation in fixed point and floating point format
- nine independent arithmetic units
- twelve word instruction stack
- synchronous internal logic with 27.5 nanosecond clock period

CPU small core memory
- 65,536 words of coincident current memory (60 bit)
- 32 independent banks
- 2048 words per bank
- 275 nanosecond read/write cycle time
- 27.5 nanosecond per word maximum transfer rate

CPU large core memory
- 512,000 words of linear select memory (60 bit)
- 8 independent banks
- 64,000 words per bank
- 1760 nanosecond read/write cycle time
- 8 words read simultaneously each reference
- 27.5 nanosecond per word maximum transfer rate

CPU input-output section
- 15 independent channels (asynchronous)
- each channel full duplex (60 bit)
- buffer areas of 64 or 128 words each channel
- 55 nanosecond per 60 bit word maximum transfer rate

PPU computation section
- 12 bit internal word
- binary computation in fixed point
- synchronous internal logic with 27.5 nanosecond clock period

PPU core memory
- 4096 words of coincident current memory (12 bit)
- two independent banks
- 2048 words per bank
- 275 nanosecond read/write cycle time

PPU input-output section
- 8 independent channels (asynchronous)
- each channel full duplex (12 bit)
- 137.5 nanosecond per 12 bit word maximum transfer rate

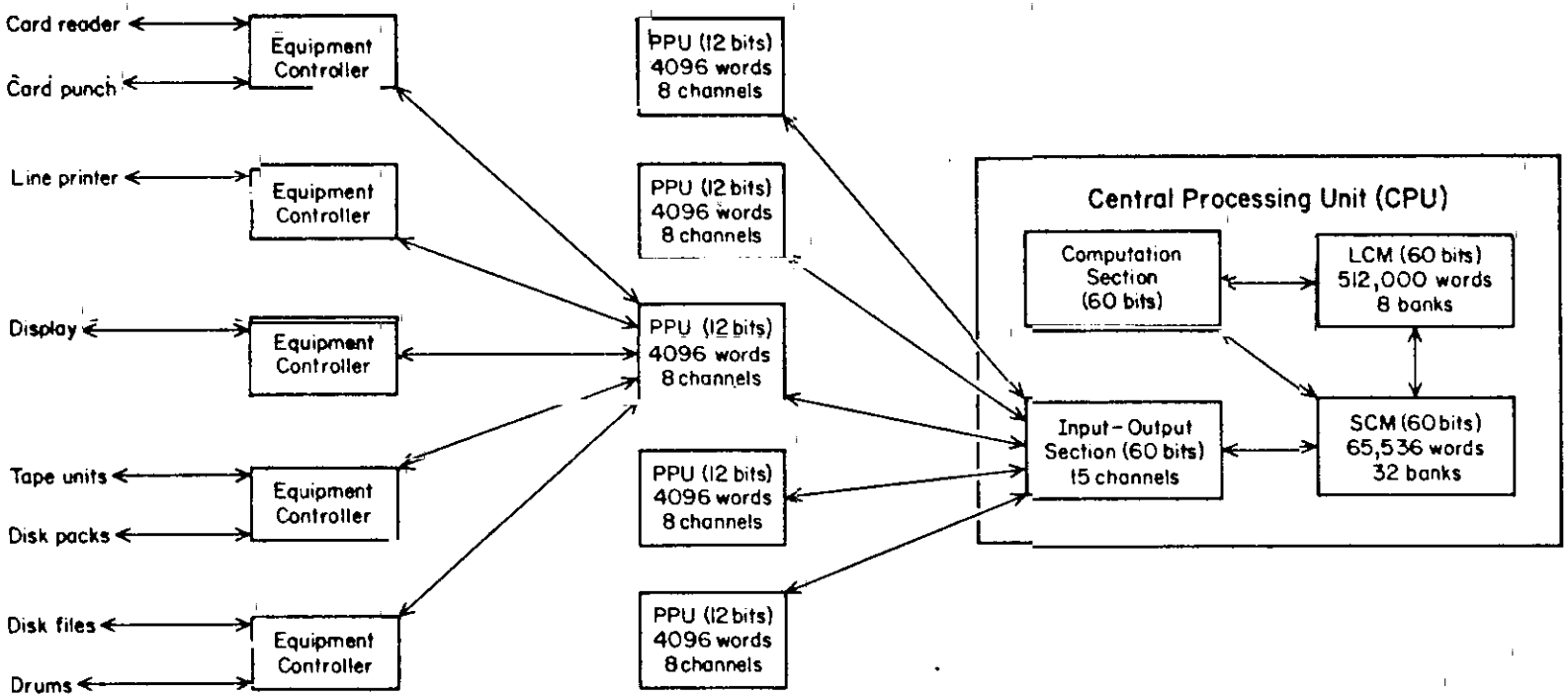


Fig. 1– 7600 System Communication

System communication

The 7600 system is divided into a number of major sections which are interconnected as shown in figure 1. All input data enters the system at a peripheral equipment controller. All output data leaves the system at a peripheral equipment controller. The PPU serve to gather input data from the peripheral equipment controllers for delivery to the CPU for processing, and distribute processed data to the equipment controllers for output devices. The communication between PPU and equipment controller is generally limited by the rate at which the equipment controller can deliver or accept data. Most equipment controllers in the 7600 system will contain a core memory buffer capable of holding one record of data for the attached input or output devices. The communication between PPU and equipment controller is then a one record burst of data followed by a relatively long period of inaction. The PPU is then able to time share its channel to the CPU among a number of equipment controllers without storing more than one record of data in its own memory at any one time.

Communication between a PPU and a peripheral equipment controller is over a 12 bit full duplex channel. Each channel has a 12 bit data path from PPU to controller, and a separate 12 bit data path from controller to PPU. These two data paths are independent and may operate simultaneously. Each path has two associated control lines carrying control information in the direction of data flow. These lines carry a "word flag" to indicate passage of each 12 bit word of data, and a "record flag" to indicate the completion of a record of data. Each path has one associated control line carrying control information against the direction of data flow. This line carries a "word resume" signal to indicate receipt of a data word. These channels are described in detail in part 4 of this manual.

Communication between a PPU and the CPU is over a 12 bit full duplex channel identical to that described above. The 12 bit data path from PPU to CPU includes a 60 bit assembly register at the CPU end of the data link. This register assembles five 12 bit words into a 60 bit word for entry into the CPU memory. The 12 bit data path from CPU to PPU includes a 60 bit disassembly register at the CPU end of the data link. This register disassembles a 60 bit word from the CPU memory into five 12 bit words for transmission over the data link.

A maximum of 15 PPU may be directly connected to the CPU. Each CPU channel has assembly and disassembly registers to convert from 60 bit to 12 bit word length. All 15 CPU input-output channels may be in operation at the same time. Data is transmitted to, or from, PPU on a record by record basis. The CPU program is interrupted at the end of a record transmission to exchange control information with the communicating PPU or to initiate transmission of another record. On very long records the CPU program is interrupted at prescribed intervals in the buffer data. The frequency of this interruption is a function of the buffer size and may be preset individually for each CPU channel. Details of the CPU buffer operation are described in detail in part 2 of this manual.

Operating system

The 7600 hardware was designed with a particular software approach in mind. This approach is an outgrowth of experience with the Chippewa Operating System (COS) for the 6600 system. The 7600 operating system software will be described in a separate manual. Reference to software in this manual will be limited to those areas where hardware provisions are a direct result of software considerations.

System monitor

The system monitor is a CPU program in the 7600 system. This program is loaded with the operating system on a machine "dead start" and remains in the CPU core memory as long as the operating system is used. A portion of the system monitor resides permanently in the small core memory (SCM) section of the CPU. This portion of the monitor program is called the "resident monitor" program. The bulk of the system monitor resides in the large core memory (LCM) section of the CPU. This portion of the monitor program is called piecemeal into the SCM for execution as overlays on the resident monitor program.

Object program

An object program is defined in this manual to mean any CPU program other than the system monitor program. This term is used to describe generally a job oriented program. An object program may be a machine language program such as a FORTRAN compiler, or it may be a program which results from compiling FORTRAN statements with a compiler.

Central processing unit (CPU)

The CPU is a single integrated data processing unit. It consists of a computation section, small core memory, large core memory, and input-output section. These sections are all contained in one main frame cabinet and operate in a tightly synchronous mode with a clock period of 27.5 nanoseconds. Communication with equipment outside of the main frame cabinet is asynchronous.

CPU core memory

The CPU contains two types of internal core memory. One type, designated as the small core memory (SCM), is a many bank coincident current type memory with a total capacity of 64K words of 60 bit length (K = 1024). The other type, designated as the large core memory (LCM) is a linear selection type of memory in which eight 60 bit words are addressed as a single unit. The LCM has a total capacity of 500K words of 60 bit length. These two types of internal memory have significantly different system functions in the CPU.

The SCM is arranged in 32 banks of 2K words each. Each bank is independent of the other 31 banks. Maximum data transfer rate between the SCM as a unit and other parts of the system is one word each clock period. Each SCM bank has a four clock period access time from arrival of the storage address to readout of the 60 bit word. The total read/write cycle time for a SCM bank is ten clock periods. It is thus possible for a maximum of ten SCM banks to be in operation at one time. This maximum occurs during block copy instructions between SCM and LCM in which the addresses for sequential words cause no SCM bank conflicts. In random addressing of the SCM for CPU program data, CPU instructions, and input-output channel data, an average of four SCM banks in operation at one time is more normal.

The SCM performs certain basic functions in system operation which the LCM cannot effectively perform. These functions are essentially ones requiring rapid random access to unrelated fields of data. The first 4K addresses in SCM are reserved for input-output buffer and control areas. These areas are addressed by the CPU input-output section as required to service the communication channels to the PPU. CPU object programs do not have access to these areas. The next 1K addresses are reserved for the resident monitor program.

The remainder of the SCM is divided between fields of CPU program code and fields of data for the currently executing program.

The LCM is arranged in eight banks of 64K words each. Each bank is independent of the other seven banks. A storage reference to a LCM bank results in a read/write cycle which takes 64 clock periods. Eight 60 bit words are read simultaneously from a LCM bank whenever a read/write cycle occurs. These words are held in a 480 bit operand register for each LCM bank. Subsequent reference to a word residing in one of these operand registers allows either read or write function without the delay of a bank read/write cycle. Maximum data transfer rate between the LCM as a unit and other parts of the system is one word each clock period. This maximum transfer rate occurs during block copy instructions between LCM and SCM. LCM bank read/write cycles are anticipated in the block copy operation to avoid a bank access delay.

The LCM provides the basic working storage for the CPU. All object programs are assembled here for execution in the SCM. All data files are buffered through LCM for the object programs. Small object programs are generally run to completion in SCM with the complete input file in LCM at the beginning of execution, and the complete output file in LCM at the end of execution.

The low order addresses in LCM are reserved for monitor program overlays, mathematic routine library, and FORTRAN compiler. These areas require approximately 32K of the 500K available storage. The remainder of LCM is divided into fields for the various operating stations in the system.

Computation section

The computation section of the CPU contains nine segmented arithmetic units, 24 operating registers, and a 12 word instruction stack. These units work together to execute a CPU program stored in the SCM. Data moves into, and out of, the computation section of the CPU through the operating registers. Data may be directly addressed in either the SCM or the LCM. The general information flow in this section is illustrated in figure 1-2.

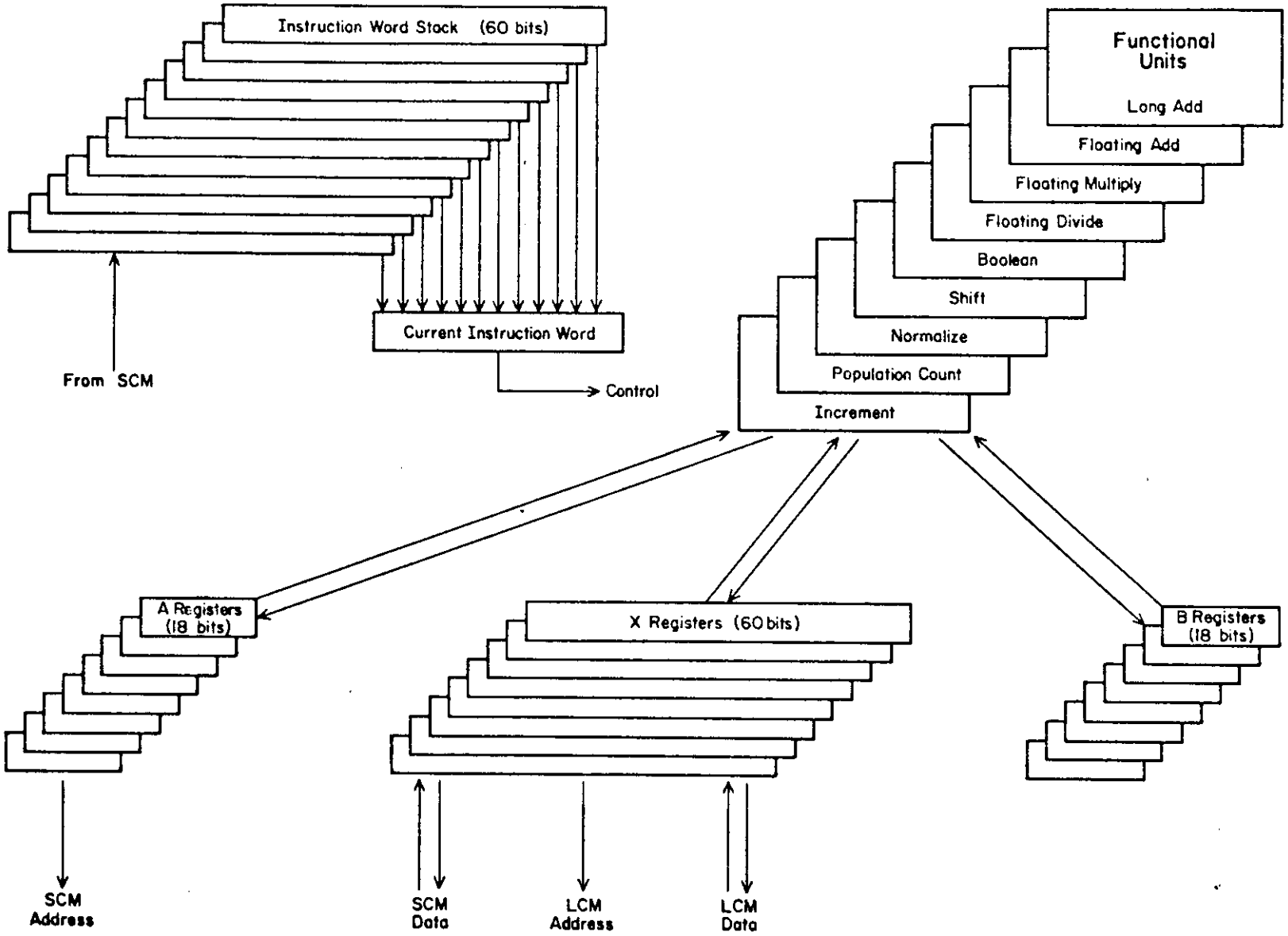


Fig. I–2 CPU Computation Section

Instruction word stack

The instruction word stack is a group of twelve 60 bit registers in the CPU computation section which hold program instruction words for execution. The instruction stack information is essentially a moving window in the program code. The stack is filled two words ahead of the program address currently being executed. A small program loop may frequently be entirely contained within the instruction stack. When this happens the loop may be executed repeatedly without further references to SCM.

The current instruction word is contained in a special register in the CPU computation section. This register is designated the CIW register. Program instruction words are read one at a time from the instruction stack and are interpreted in the CIW register for execution. This register controls all of the data transmission paths between the operating registers and the functional units in the computation section.

X registers

There are eight 60 bit X registers in the computation section of the CPU. These registers are the principal data handling registers for computation. Data flows from these registers to the SCM and the LCM. Data also flows from SCM and LCM into these registers. All 60 bit operands involved in computation must originate and terminate in these registers. Detail characteristics of these registers are provided in part two of this manual.

A registers

There are eight 18 bit A registers in the computation section of the CPU. These registers are essentially SCM operand address registers. These registers are associated one-for-one with the X registers. When an address is entered in an A register the corresponding data in the X register is normally read from, or stored into, SCM at that address.

B registers

There are eight 18 bit B registers in the computation section of the CPU. These registers are primarily indexing registers for controlling program execution. Program loop counts may be incremented or decremented in these registers. Program addresses may be modified on the way to an A register by adding or subtracting one or more B register quantities. Further detail on these registers is provided in part two of this manual.

Functional units

There are nine functional units in the computation section of the CPU. Each is a specialized arithmetic unit with algorithms for a portion of the CPU instructions. Each unit is independent of the other units, and a number of functional units may be in operation at the same time. There are no "visible" registers in the functional units from a programing standpoint. A functional unit receives one or two operands from operating registers at the beginning of instruction execution and delivers the result to the operating registers when the function has been performed. There is no information retained in a functional unit for reference in subsequent instructions. These units operate essentially in a three address mode, with very limited source and destination addressing.

All functional units with the exception of the floating multiply and divide units have one clock period segmentation. This means that the information arriving at the unit, or moving within the unit, is captured and held in a new set of registers at the end of every clock period. It is therefore possible to start a new set of operands for unrelated computation into a functional unit each clock period even though the unit may require more than one clock period to complete the calculation. This process may be compared to a delay line in which data moves through the unit in segments to arrive at the destination in the proper order but at a later time. All functional units perform their algorithms in a fixed amount of time. No delays are possible once the operands have been delivered to the front of the unit.

The floating multiply unit has two clock period segmentation. Operands may enter the multiply unit in any clock period providing there was no operation initiated in the preceding clock period. There is a one clock period delay in initiating a multiply instruction if another multiply instruction has just been started.

The floating divide unit is the only functional unit in which an iterative algorithm is executed. There is essentially no segmentation possible in this unit although the beginning of a new operation can overlap the completion of the previous operation by two clock periods.

The table below lists the functional units with the number of clock periods required for execution in each unit. The first column indicates the number of clock periods in each segment. The second column indicates the number of clock periods required to execute the function from the time the operands leave the operating registers until the result arrives back at the operating registers. Each functional unit has a fixed execution time which is independent of its possible modes of operation. For example, a double precision multiply requires the same amount of time as a single precision multiply. The list of octal designators for each functional unit identifies which CPU instructions are performed in that unit.

	segment time	execution time
Long add unit (36, 37)	1 clock period	2 clock periods
Floating add unit (30, 31, 32, 33, 34, 35)	1 clock period	4 clock periods
Floating multiply unit (40, 41, 42)	2 clock periods	5 clock periods
Floating divide unit (44, 45)	18 clock periods	20 clock periods
Boolean unit (10 through 17, 26, 27)	1 clock period	2 clock periods
Shift unit (20, 21, 22, 23, 43)	1 clock period	2 clock periods
Normalize unit (24, 25)	1 clock period	3 clock periods
Population count unit (47)	1 clock period	2 clock periods
Increment unit (50 through 77)	1 clock period	2 clock periods

Binary arithmetic

All binary arithmetic operations in the CPU computation section are performed in a ones complement subtractive mode. This is called simply "ones complement" mode in the remainder of this manual. This mode of arithmetic is represented by the recursive boolean expressions below for the sum of two binary numbers.

```
A(I) = .NOT. C(I) .AND..NOT. D(I) .AND..NOT. B(I) .OR.
       C(I) .AND. D(I) .AND..NOT. B(I) .OR.
       C(I) .AND. B(I) .AND..NOT. D(I) .OR.
       D(I) .AND. B(I) .AND..NOT. C(I)

B(I+1) = .NOT. C(I) .AND..NOT. D(I) .AND..NOT. B(I) .OR.
         B(I) .AND..NOT. C(I) .AND..NOT. D(I) .OR.
         C(I) .AND. B(I) .AND..NOT. D(I) .OR.
         D(I) .AND. B(I) .AND..NOT. C(I)

B(0) = B(M)
```

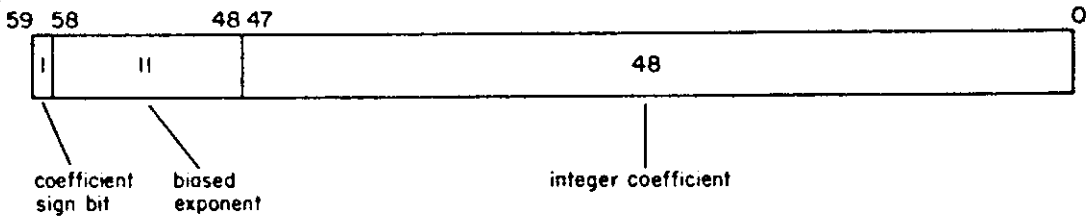
where: M = number of bit positions in adder
D(I) = addend bit I
C(I) = augend bit I
B(I) = borrow into bit position I
A(I) = sum bit I
I = 0,1,2,...,M-1

The above expressions for the addition of two integers are symmetrical in the appearance of the augend and addend bits. The order of addition is therefore not important. This form of arithmetic creates two representations of zero. A word of all zero bits is positive zero. A word of all one bits is negative zero. A negative zero can be generated in the addition process only if the addend and the augend are negative zero. The modulus in the addition process is always the power of two corresponding to the adder length, minus one.

Subtraction is performed by complementing the subtrahend and adding to the minuend. Multiplication and division are sequences of addition operations.

Floating point arithmetic

Floating point numbers are represented in a standard format throughout the CPU. This format is a packed representation of a signed binary integer coefficient times two with a signed binary integer exponent. The coefficient is a 49 bit ones complement integer. The exponent is an eleven bit ones complement integer. The sign of the coefficient is separated from the rest of the coefficient as shown in the 60 bit word organization below.



Floating point format

The exponent portion of the floating point format is biased by complementing the exponent sign bit. This particular format for floating point numbers is chosen because the packed form may be treated as a 60 bit integer for sign, threshold, equality, and zero tests. The same set of branch instructions can therefore be used for both integer and floating point forms. A threshold test can be made by subtracting two floating point numbers in the long add unit, rather than the floating add unit, thus saving two clock periods in execution.

Some examples of packed and unpacked floating point numbers are shown below in octal notation to illustrate the packing process. The first two examples are different forms of the integer +1. The third example is +100 decimal and the fourth example is -100 decimal. The last two examples are of very large and very small positive numbers.

```
unpacked coefficient = 0000 0000 0000 0000 0001
unpacked exponent = 00 0000
packed format = 2000 0000 0000 0000 0001

unpacked coefficient = 0000 4000 0000 0000 0000
unpacked exponent = 77 7720
packed format = 1720 4000 0000 0000 0000

unpacked coefficient = 0000 6200 0000 0000 0000
unpacked exponent = 77 7726
packed format = 1726 6200 0000 0000 0000

unpacked coefficient = 7777 1577 7777 7777 7777
unpacked exponent = 77 7726
packed format = 6051 1577 7777 7777 7777

unpacked coefficient = 0000 4771 3000 0044 7021
unpacked exponent = 00 1363
packed format = 3363 4771 3000 0044 7021

unpacked coefficient = 00 6301 0277 4315 6033
unpacked exponent = 77 6210
packed format = 0210 6301 0277 4315 6033
```

Special floating point forms

Special values are used in floating point format to indicate overflow of the floating point range, underflow of the floating point range, and indefinite results. These special values are sensed by the functional units to preserve the significance of the calculation as long as possible.

Overflow of the floating point range is indicated by an exponent value of +1777 octal. This is the largest exponent value that can be represented in the floating point format. This exponent value may result from the calculation in a floating point unit in which this exponent value, together with the computed coefficient value, is a correct representation of the result. This situation is called a "partial overflow" in this manual. An overflow error condition is not indicated by the functional unit generating this result. Further computation in floating point functional units using this result will be detected as an overflow, however. A "complete overflow" occurs whenever a floating point functional unit computes a result which requires an exponent larger than +1777 octal. In this case the functional unit indicates an overflow error condition and packs a "complete overflow" value for the result. This result has a +1777 exponent and a zero coefficient. The sign of the coefficient will be the same as that which would have been generated if the result had not overflowed the floating point range.

Underflow of the floating point range is indicated by an exponent value of -1777 octal. This is the smallest exponent value that can be represented in the floating point format. This exponent value may result from the calculation in a floating point unit in which this exponent value, together with the computed coefficient value, is a correct representation of the result. This situation is called a "partial underflow" in this manual. An underflow error condition is not indicated by the functional unit generating this result. Further computation in floating point functional units using this result may be detected as an underflow, however. A "complete underflow" occurs whenever a floating point functional unit computes a result which requires an exponent smaller than -1777 octal. In this case the functional unit indicates an underflow error condition and packs a "complete underflow" value for the result. This result has a -1777 exponent and a zero coefficient. The sign of the coefficient will be the same as that which would have been generated if the result had not underflowed the floating point range. The complete underflow indicator is a word of all zero bits, or all one bits, depending on the sign. It is the same as a zero word in integer format.

An indefinite result indicator is generated by a floating point functional unit whenever the calculation cannot be resolved. This is the case in division when the divisor is zero and the dividend is also zero. It is also the case in multiplication of an underflow number times an overflow number. The indefinite result indicator is a value that cannot occur in normal floating point calculations. This indicator corresponds to a minus zero exponent and a zero coefficient. An indefinite error condition is indicated by the functional unit generating this result. Any floating point functional unit receiving an indefinite indicator as an operand will generate an indefinite result no matter what the other operand value. Indefinite indicators are always generated with a positive sign. They may occur as operands with negative sign, however, because of complementation in the boolean unit.

Normalized floating point

A floating point number in packed format is normalized if the coefficient sign bit is different from bit 47. This condition implies that the coefficient has been shifted to the left as far as possible, and therefore the floating point number has no leading zeros in the coefficient. The normalize unit performs this function. The floating multiply and floating divide units deliver normalized results when provided with normalized operands. The floating add unit may deliver un-normalized results even if both operands are normalized. It is therefore necessary to program the normalize operation in the normalize unit after each sequence of floating add or subtract operations if the result is to be kept in a normalized form.

Double precision numbers

Computation in double precision or multiple precision modes may be performed with the aid of the double precision instructions (32, 33, 42). The floating add unit and the floating multiply unit perform all computation in a double precision mode. The single precision instructions use only the upper half of the 96 bit result. The double precision instructions perform the same calculation but deliver the lower half of the 96 bit result with the appropriate exponent value. Double precision division must be programed using the single precision divide instruction as a first approximation. It is necessary to reconstruct the remainder by multiplying quotient by divisor in a double precision mode and subtracting from dividend.

Rounded computation

Optional floating point instructions are provided to round the results in single precision computation. These instructions are executed in the same amount of time as the unrounded versions. The operands are modified in the functional units to accomplish the rounding function. The amount of bias introduced by the rounding operation varies from unit to unit and is affected by the coefficient value in the operands. These effects are described in detail for each of the round instructions in part 3 of this manual.

Integer multiplication

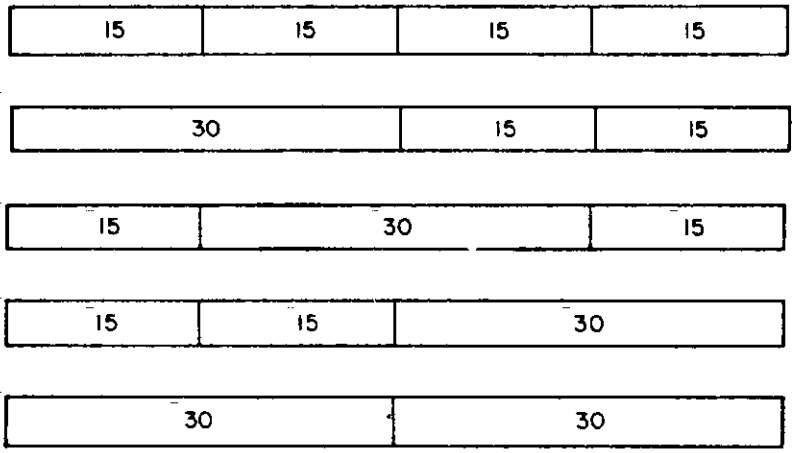
There is no CPU integer multiply instruction. Integer multiplication must be performed in the floating multiply unit. This is accomplished by packing the integers into floating point format using the pack instruction and a zero exponent value. The product can be formed for small integers without normalizing the operands by using the double precision multiply instruction. The result need not be unpacked if the destination for the product is an A register or a B register since the increment unit extracts only the lowest order 18 bits of the 60 bit word.

Integer division

There is no CPU integer divide instruction. Integer division must be performed in the floating point divide unit. This is accomplished by packing the integers into floating point format using the pack instruction and a zero exponent value. The divisor must then be normalized with a 24 instruction. The dividend need not be normalized. The resulting quotient must be unpacked and the coefficient shifted by the amount of the unpacked exponent using the 22 instruction to obtain the integer quotient.

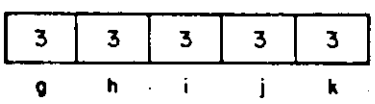
CPU instruction format

Program instruction words are divided into four 15 bit fields called "parcels" in this manual. The first parcel in a word is defined to be the highest order 15 bits of the 60 bit word. The second, third, and fourth parcels then follow in order. A CPU instruction may occupy either one or two parcels, depending on the type of instruction. If an instruction requires two parcels it must not begin in the fourth parcel of the word. The possible arrangements of one and two parcel instructions in a 60 bit word are shown below.



It may be necessary in program code to occasionally complete a 60 bit instruction word with one parcel pass instructions. This must be done in those cases where a two parcel instruction would require starting in the fourth parcel of a word. One parcel pass instructions are also used to complete a 60 bit word in order to place a particular instruction in the first parcel of a word. This is necessary for branch entry points because a branch instruction destination address must begin with a new word.

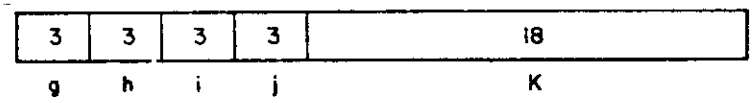
A one parcel instruction is composed of five octal digits called "designators" in this manual. These designators are identified by the symbols g, h, i, j, and k. The designators are arranged in order in the 15 bit parcel as shown below.



One parcel instruction format

The g designator generally identifies the type of instruction and frequently specifies the functional unit. The h designator completes the function code specification for all but a few instructions by specifying the functional unit mode. The i, j, and k designators are the operand source and destination indicators. These designators specify which one of the eight possible A, B, or X registers is referenced. The selection of register type--A, B, or X--is implied in the instruction function code by the g and h designators. The i designator is always the destination indicator. If there are two destinations required for the instruction, both the i and j designators specify destination.

A two parcel instruction contains an 18 bit operand to be used in the instruction execution. This is used for branch destination addresses and for small integer constants. There are five designators in this instruction format. The g, h, i, and j designators have similar roles to those described for the one parcel instructions. The k designator is expanded into the 18 bit K operand as shown below.



Two parcel instruction format

The two parcel instructions should be used only where the operand in the instruction is an invariant throughout the complete execution of the program in SCM. Modification of program code during execution of that code has several problems which are not easily covered in program layout. There are no hardware provisions to update the content of the instruction stack, for example, when one of the instructions in the stack is modified in storage. The two parcel instructions are executed in the same time as equivalent one parcel instructions. Two parcel instructions tend to accelerate the program code movement through the instruction stack, however, which causes increased storage references for program code and more frequent delays in filling the instruction stack. As a result, the one parcel instructions should be favored whenever a choice is available in program coding.

Storage field protection

Each object program at execution time has a designated field of SCM and a designated field of LCM in which it may address data. These fields are specified by the monitor program at the time the object program is initiated. Each field may begin at an arbitrary address in storage and continue for an arbitrary length. All addresses in an object program field must be contiguous.

The storage bounds for an object program are contained in four hardware registers in the CPU. These registers are:

(RAS) Reference address for small core memory
This is an 18 bit register which defines the absolute SCM address which is the first address in the SCM field.

(FLS) Field length for small core memory
This is an 18 bit register which defines the length of the SCM field.

(RAL) Reference address for large core memory
This is a 24 bit register which defines the absolute LCM address which is the first address in the LCM field.

(FLL) Field length for large core memory
This is a 24 bit register which defines the length of the LCM field.

All addresses for SCM or LCM contained in the object program code are relative to the reference address which begins the defined field. It is therefore not possible for an object program to read or alter any storage locations with a lower absolute address than the reference address. Each object program reference to storage is checked against the appropriate field length to determine if the address is within the bounds assigned. A storage reference beyond the assigned field length is prevented from altering the storage content and creates an error condition which terminates the program execution.

Program branching

Program branching presents some special situations because of the instruction stack. The current program address is maintained at all times in a program address register (P). This register contains the relative address in SCM for the word currently in the CIW register. When program instruction words are read sequentially from the instruction stack into the CIW register, the P register is advanced one count for each word. The instruction stack contains 12 words of instruction code in a group of registers called the instruction word stack (IWS). Associated with each word in the instruction word stack is an 18 bit address. The 12 addresses in the instruction stack are contained in twelve 18 bit registers called the instruction address stack (IAS). The addresses in the IAS move with the words in the IWS so that the one-for-one relationship between address and program instruction word is maintained as the words move through the instruction stack.

When a program branch point is reached and the current program sequence is terminated with a jump to a new program address, the new program address is entered in the P register. This new program address is then compared with the 12 addresses in the IAS to determine if the jump is within the instruction stack. If a coincidence is found between the new address in the P register and one of the 12 addresses in the IAS, the associated word in the IWS is read immediately into the CIW register and the jump instruction has been executed. If no coincidence is found between (P) and the addresses in the IAS, the jump is "out of stack." In this case a new sequence of instructions must be read from SCM at the new program address (P) and entered in the instruction stack. Completion of the jump instruction is delayed in this case by the amount of time required for the first word to be read from SCM.

The old instruction words in the IWS are not cleared when a jump out of stack occurs. The old words are simply shifted along in the IWS as the new program sequence enters the instruction stack. It is therefore possible to have several sequences of noncontiguous program code in the instruction stack at one time. The program execution may jump back and forth between these program sequences without leaving the instruction stack as long as the current program address (P) does not come within two words of the end of the program sequences held in the instruction stack.

Exchange jump

The CPU exchange jump is a mechanism for switching CPU execution between object program and monitor program. An object program which requires monitor action for a library call, input-output request, or error treatment performs an exchange jump to terminate its own execution and begin the monitor program. Similarly, the monitor program performs an exchange jump to initiate execution of a particular object program.

The execution of an exchange jump involves the simultaneous storing of all pertinent information in the CPU operating registers and control registers into SCM, and the reading of new information from SCM into these same registers. This block of data is called an "exchange package." The execution of an exchange jump then involves the storing of the exchange package for the terminating program and the reading of the exchange package for the initiating program. The information contained in an exchange package is shown in figure 3 on the following page.

An "execution interval" for an exchange package is defined in this manual to mean a period of time during which the particular exchange package resides in the CPU hardware registers. The execution interval begins with an exchange jump which reads the exchange package from SCM and enters these parameters in the CPU registers. The execution interval ends with another exchange jump which stores the exchange package back into SCM. The complete execution of an object program may then be composed of a number of execution intervals for the object program exchange package interspersed with monitor activity.

An exchange package contains all of the information necessary to resume the execution of a terminating program. The contents of the operating registers A, B, and X are contained in the exchange package along with the current program address P. The four storage bounds registers RAS, FLS, RAL, and FLL are represented. In addition the following four registers of special information are included in the exchange package.

(NEA) Normal exit address - This is a SCM absolute address for an object program exchange exit instruction.

(EEA) Error exit address - This is a SCM absolute address for an exchange jump on error termination.

(BPA) Breakpoint address - This is a SCM relative address for breakpointing an object program.

(PSD) Program status designation - This is a register of control information.

SCM location				
	n	P	A0	BPA
	n+1	RAS	A1	B1
	n+2	FLS	A2	B2
	n+3	PSD	A3	B3
	n+4	RAL	A4	B4
	n+5	FLL	A5	B5
	n+6	NEA	A6	B6
	n+7	EEA	A7	B7
	n+8	X0		
	n+9	X1		
	n+10	X2		
	n+11	X3		
	n+12	X4		
	n+13	X5		
	n+14	X6		
	n+15	X7		

Fig. 1-3 CPU Exchange Package

Program breakpoint

An object program may be executed in small sections during a debugging phase by using the breakpoint address register (BPA). This is a hardware register in the computation section of the CPU which is loaded from the object program exchange package. A coincidence test is made between (BPA) and the program address register (P) as each program instruction word is read from the IWS to the CIW register. When a coincidence occurs the program execution is terminated with an exchange jump to the error exit address (EEA).

The monitor program controls the breakpoint address for debugging an object program by altering the exchange package for the object program before each execution interval. The monitor program receives instructions for breakpoint control from an operator console or from control cards in a job stack. It is possible to step through a program one instruction word at a time using an operator console to monitor the register values at each step.

Error exits

Execution of an object program may be terminated by an exchange jump to the error exit address (EEA) under certain conditions. Some of these conditions may be selected by mode declarations through the monitor program, and some are unconditional. In general, errors due to arithmetic overflow, underflow, or indefinite results during computation may be allowed to proceed through the calculation, or may cause an error exit, depending on mode selection. Errors due to hardware failure or program addressing out of an assigned field in storage cause unconditional error exits. In any error exit case the monitor program has the ability to continue the object program where the error can be corrected or ignored.

The error condition flags and mode selection flags are all contained in an 18 bit program status designation (PSD) register. This register is loaded from the exchange package for each object program. The mode selections are made in the exchange package prior to the execution interval by the monitor program. If an error condition occurs during the execution interval the monitor program can determine the type of error by analyzing the terminating exchange package parameters. Each bit in the PSD register has significance either as a mode selection or an error condition flag. These flags are described in detail in part 2 of this manual.

CPU input-output section

The CPU input-output section includes the mechanism to buffer data to (or from) the directly connected PPU. Each PPU communicates with the CPU over a 12 bit full duplex channel. Each channel has assembly and disassembly registers to convert the 12 bit channel data to 60 bit CPU words. The function of the CPU input-output section is to deliver these 60 bit words to the SCM for incoming data, read 60 bit words from the SCM for outgoing data, and interrupt the CPU program for monitor action on the buffer data as required.

The input-output section is able to process a maximum of one 60 bit word each two clock periods. The effective processing rate is somewhat lower than this because of bank storage conflicts in SCM. Whenever a bank conflict occurs on an input-output section request, the communication path to the SCM is held up until the conflict is resolved. Channel requests for a SCM word reference are processed on a priority basis whenever the I/O section is not able to keep up with the channel requests. The priority is assigned in order by channel number, with the lowest order channels having the highest priority.

There are a total of 15 channels in the I/O section of the CPU. These channels are numbered in octal beginning with 01 and ending with 17. Each channel has a SCM buffer area for incoming data and a separate SCM buffer area for outgoing data. In addition each channel has an exchange package for incoming data and an exchange package for outgoing data. Each buffer area is divided into two fields, a lower field and an upper field. Data is entered (or removed) from the buffer area in a circular mode. The last word in the lower field is followed by the first word in the upper field. The last word in the upper field is followed by the first word in the lower field. Whenever a buffer area has been filled (or emptied) to the point where a field boundary is crossed, the CPU is interrupted through the associated exchange package to process the buffer data. The channel continues to fill (or empty) the other buffer field while the CPU is processing this buffer data. For further details on this buffer operation see part 2 of this manual.

The I/O section exchange package areas are permanently assigned in the lowest order addresses of SCM. These areas are arranged as shown in figure 1-4. The I/O section buffer areas are assigned in the next higher order address positions of SCM. These areas may be changed both in size and order (wiring change) to accommodate various types of channel volume. A typical arrangement for the buffer areas is shown in figure 1-5. Total I/O section space in SCM cannot exceed absolute address 10,000 octal.

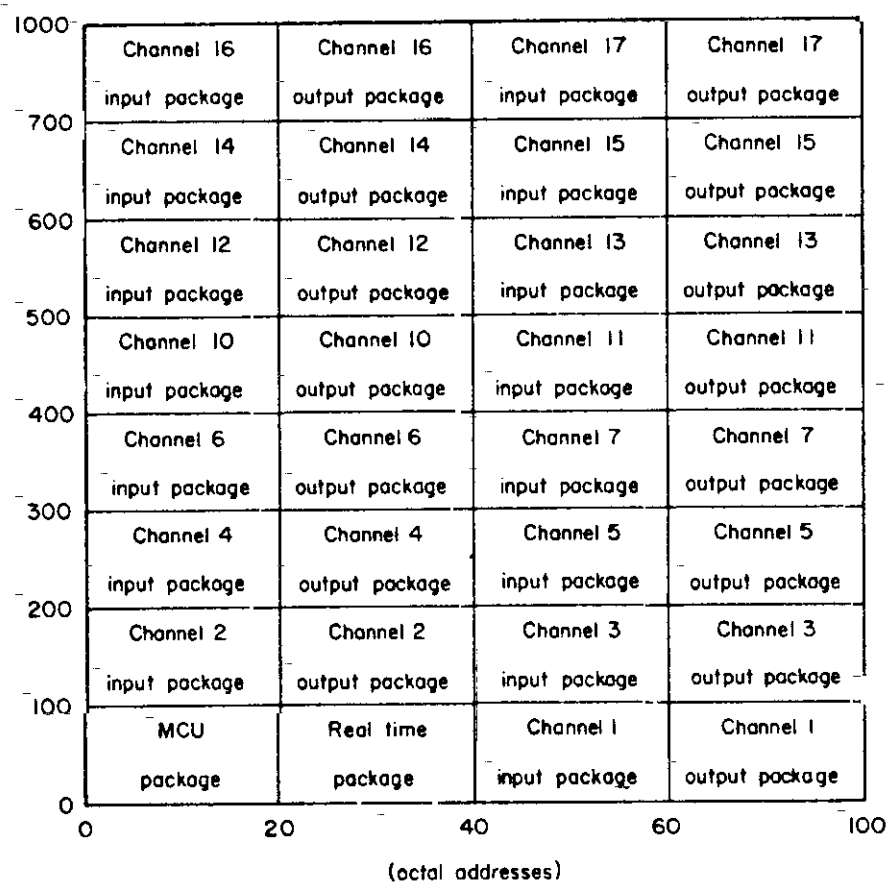


Fig. 1-4 I/O Section Exchange Package Areas In SCM

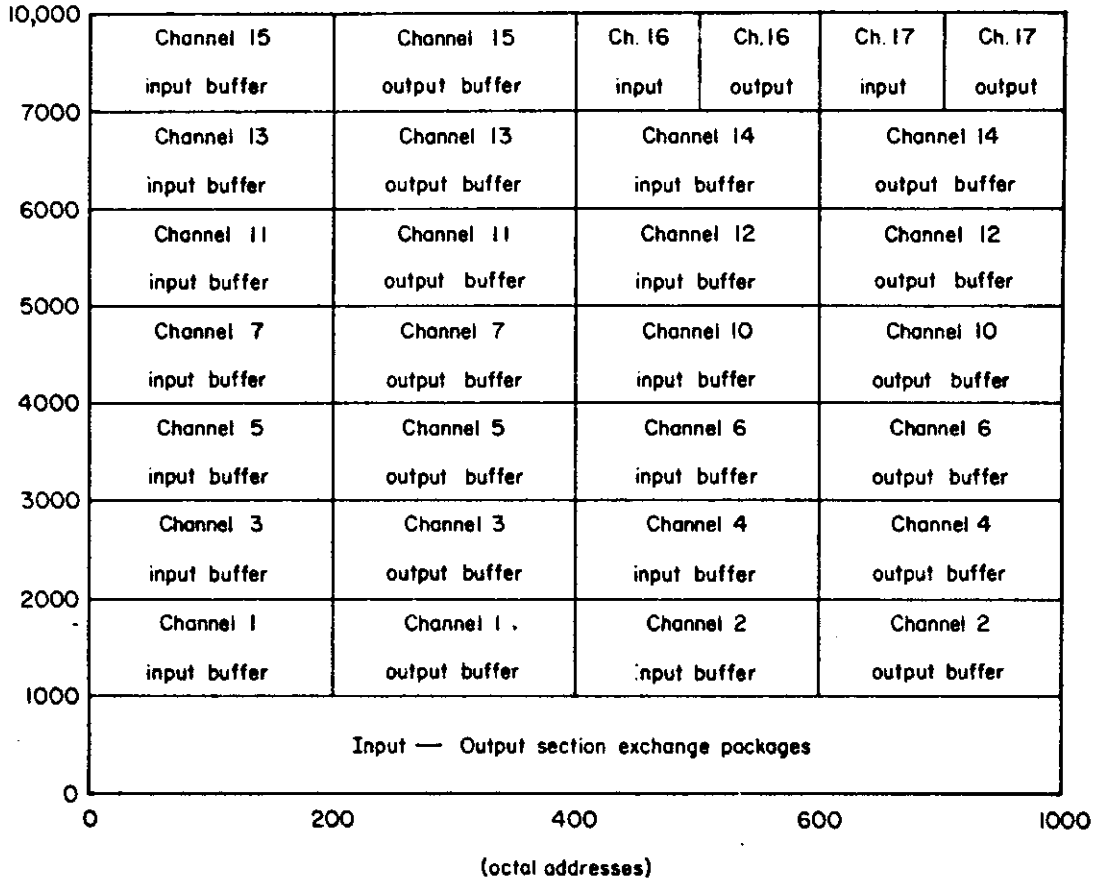


Fig. 1–5 I/O Section Buffer Areas In SCM

Real time clock

CPU programs may be timed precisely by using the CPU clock period counter. This counter is essentially a real time clock which is advanced one count each clock period of 27.5 nanoseconds. Since the clock is advanced synchronously with the program execution, the program may be timed to an exact number of CPU clock periods.

The CPU clock period counter contains a 17 bit register which can be read by a CPU program with an 016 instruction. This register contains the lowest order 17 bits of the real time count. An overflow of the highest order bit in this register sets a real time interrupt flag. This flag reads as an 18th bit on an 016 instruction. In addition, this flag causes an exchange jump to the real time exchange package at absolute address 0020 in SCM.

Real time interrupt

The CPU clock period counter causes an interrupt of the CPU program every 3.6 milliseconds (approx.). This corresponds to the modulus of the 17 bit register in the clock period counter. This interrupt causes an exchange jump to absolute address 0020 in SCM. The real time exchange package at this SCM address executes a CPU program in monitor mode which advances the count in a 60 bit SCM storage location to continue the real time clock function of the clock period counter. This program also tests time limit for the currently active object program. The real time interrupt flag is cleared at the end of the execution interval for real time exchange package. This flag is read as an 18th bit on an 016 instruction in order to avoid erroneous timing indication where the reading of the clock period counter coincided with the setting of the interrupt flag. This bit is an indication that the counter has passed its modulus but the interrupt to advance the SCM word has not yet occurred.

External interrupt

The CPU computation may be interrupted from an external source through the directly connected PPU. Each such PPU has the ability to interrupt the execution of an object program and call the system monitor by sending a control message over the associated channel to the CPU. The interrupt occurs whenever a record flag arrives at the CPU input-output section on an incoming data link. Interpretation of message content is a function of the system monitor program.

System dead start

The system is initially started through the maintenance control unit (MCU). This mechanism is used whenever power is turned on after an idle period or when the system is restarted after hardware failure. The MCU is essentially a PPU with specially adapted input-output channels. This unit is used exclusively for maintenance functions and is described in part 6 of this manual.

The dead start sequence begins with a deck of binary cards for the MCU program. These cards are loaded through the MCU card reader and activate the MCU. The MCU program then dead starts the CPU and all other PPU in sequence. A bootstrap program is entered directly into SCM from the MCU. The CPU program is then initiated by the MCU through the MCU exchange package at absolute address zero in SCM. This bootstrap program transmits a resident PPU program to all directly connected PPU to initiate their activity. The system is then loaded completely from a system library tape associated with one of the system PPU. This library tape may be any tape in the system and may be declared at dead start time through the maintenance console.

System operation

The operating system software consists of the CPU monitor program, with its overlays, plus the PPU programs to drive the peripheral equipment. The operating system forms a software framework in SCM and LCM to hold the object programs for execution. Organization of CPU storage is illustrated in figure 1-6 on the following page. The I/O section storage areas are fixed by hardware addressing in SCM. The remaining areas are strictly software organization.

The resident monitor program resides immediately above the I/O section in SCM. This program is a permanent part of the operating system and is never moved during system execution. This program handles all I/O section interrupt requests as well as object program requests. The remaining portion of SCM beginning at octal address 12,000 and continuing to 200,000 is available for object program code and data.

The low order addresses in LCM are used for permanent storage of frequently used programs and overlays. The system tables are kept here and are directly addressed by the monitor program. Library routines and compiler code may vary with installation requirements. Approximately 100,000 octal words of LCM are expected to be used for these permanent storage requirements.

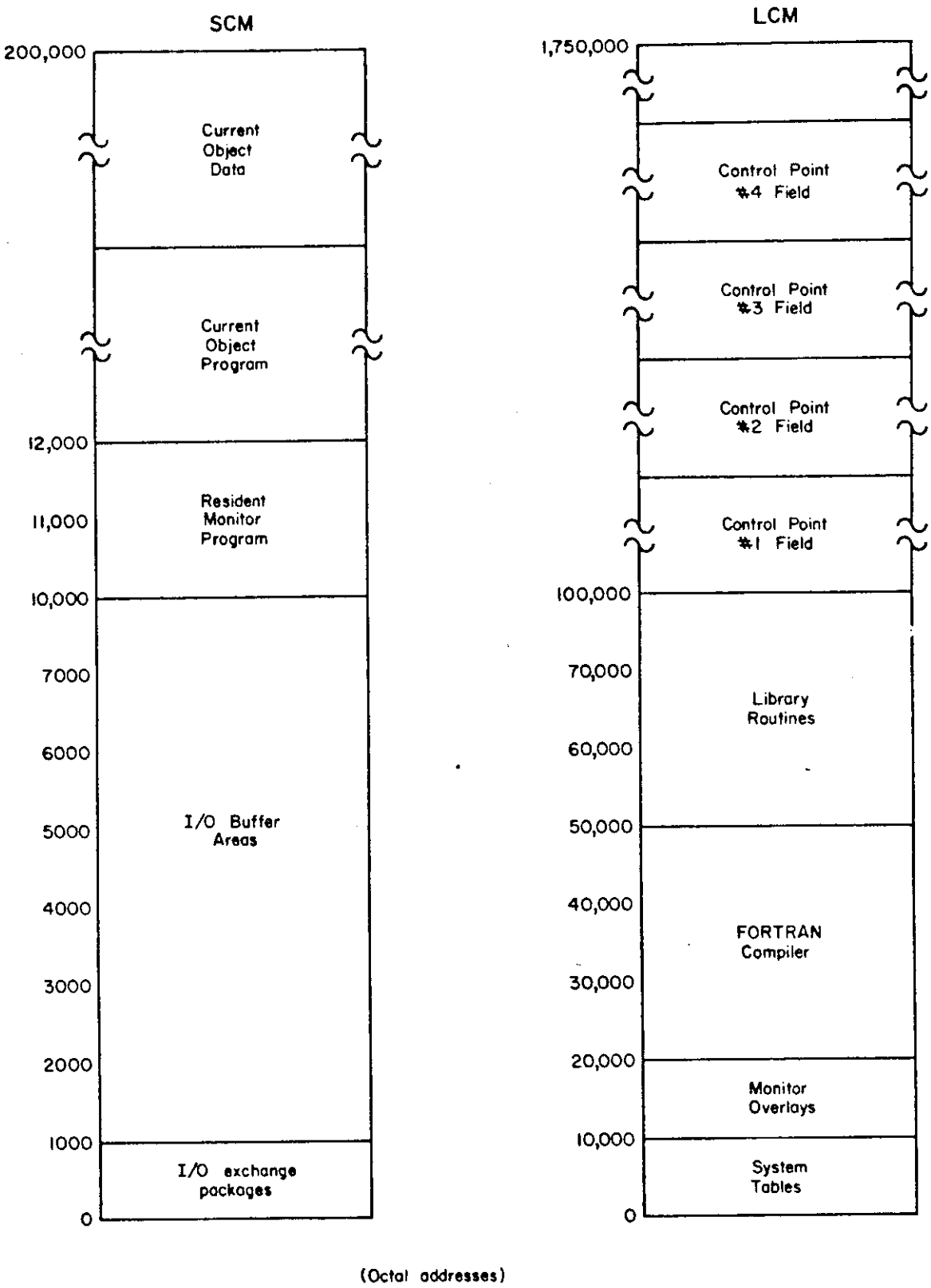


Fig. 1-6 Operating System Storage Allocation

The remainder of LCM beginning at octal address 100,000 and continuing to address 1,750,000 is available for object program code and data. This area is divided into a number of control point fields. Each control point field contains a single job oriented program code and data. In addition, each control point contains the necessary control information for continuity from one program to the next. These areas vary in size as required for each job. When a job is completed and a new job is assigned to a control point, the storage areas are readjusted by moving the data in each control point field through the SCM as a buffer to a new LCM location.

Job execution proceeds through the system in three phases. In the first phase cards are read at an operating station and an input file is generated on a disk pack, tape unit, or disk file. This input file may physically reside at the operating station or it may reside in a central disk file.

In the second phase the input file is copied into a control point field in LCM. If the input file is small the entire file may then reside in LCM. If the input file is large the first portion of the file is copied into a buffer area in the control point field. The control cards in the input file are then interpreted by the monitor program and the necessary compiler or library routines are read from outside the LCM if necessary for program execution. When the control point information is ready for execution the program code is transferred to SCM. If the program and associated data are too large to fit entirely in SCM a portion of the data must be retained in LCM and directly addressed there. This must be done by declaration at compile time in order to designate certain arrays of data to reside in LCM for execution.

Only one program at a time is executed in SCM. The entire SCM object program field may thus be used for each program. Data is read from the input file in LCM and results are stored in an output file in LCM. If the amount of input and output data is small the job may be run to completion in one execution interval. If job execution is delayed by buffer size or by intermediate file references the program code is returned to LCM and another control point uses the SCM while buffer data is transferred to (or from) LCM. The second phase is completed when the output data has been delivered to the output file buffer in LCM and this buffer has been emptied onto a disk pack, tape, or disk file for listing.

The third phase consists of copying the output file from the magnetic storage to a printer at the operating station. During this phase the LCM control point field has been released for another job.

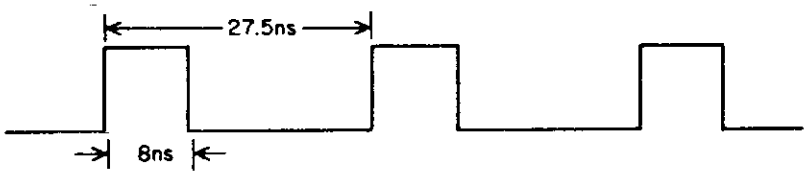
2 CPU DESCRIPTION

PART 2: CPU DESCRIPTION

Introduction

This part of the reference manual describes the various units of the CPU in detail. Each unit is described from the standpoint of its logical functions and the time relationships between signals arriving and departing to other units. Electrical pulses travel between units at approximately 7.5 inches per nanosecond. This is the propagation rate on twisted pair wires of the type used in interchassis wiring for the main frame cabinet. This travel time is a major factor in execution time for the functional units.

All timing within the main frame cabinet is controlled by a single phase synchronous clock network. This clock has a period of 27.5 nanoseconds and a pulse duration of 8 nanoseconds. The signal on a clock line would therefore appear as in the waveform below.



CPU clock waveform

The lines which carry the clock timing information from the central clock pulse source to the individual units of the CPU are all made of uniform length so that the leading edge of a clock pulse arrives at all parts of the CPU cabinet at the same time. This network of timing signals is then the framework for all interunit communication.

The individual units in the CPU are composed of "registers" and "static networks." The static networks form the logical combinations necessary to perform the desired computation. The registers serve as storage locations between the logical networks. During a clock period information moves from registers along transmission paths and through static networks to the inputs of other registers. The leading edge of a clock pulse terminates a clock period. At this time the information is sampled and stored in the registers and a new clock period begins.

CPU registers are generally of the clear/enter type. That is, new information may be entered in the register at the same instant that the old information is destroyed. The information in a given register may move through a static network and return to the same register at the end of the clock period. This creates two timing limits for the hardware circuits: a minimum "short path" time, and a maximum "long path" time. The short path limit is the same as the clock pulse duration. This means that there must be no path through a static network which returns to a register input with less than an 8 nanosecond delay. The long path limit is the same as the clock period. This means that there must be no path through a static network which returns to a register with more than 27.5 nanoseconds delay.

Timing charts in this manual are organized around clock period activity. The charts list the events which are taking place as information is in transition from one set of registers to another. Generally many activities are taking place in parallel during the same clock period. The order of the listed events within a given clock period is not important since they are really occurring simultaneously.

Frequent use is made in this manual of parenthesized register names. This is a shorthand notation for the expression "the contents of register ----." For example, a timing chart listing may say "transmit (P) to the NSA register." This means, "transmit the contents of the P register to the NSA register." Extensive use is also made of subscripted designation for the A, B, and X registers. For example, a timing chart may say "transmit (Xk) to the boolean unit." This means, "transmit the contents of the X register specified by the k designator to the boolean unit."

Control information is generally held in one bit registers called "flags" in this manual. Some flags are cleared at the end of each clock period unless they are reset specifically. These bit registers are of the clear/enter type. Other flags are set in one clock period and remain set until the information is specifically cleared. Each flag type is identified individually in this part of the manual.

Most of the registers in the CPU have no special name and are not "visible" to the programer. These registers are generally cleared at the end of each clock period unless they are reset specifically with new data. Most of the named registers have clock pulse gates to allow the information in the register to remain unaltered for indefinite periods of time. These registers are simultaneously cleared and entered with new data at the end of those clock periods in which new data is specifically transmitted to them.

Control flag names

Control flag names individually identify a one bit register with control information. These one bit registers exist throughout the CPU wherever control information must be stored at the end of a clock period for use during the following clock period. A number of control flag names are abbreviated with three characters, the last of which must be F. Many data registers are also identified with three character abbreviations, but none of these have a terminal letter F.

Data register names

The A, B, X, and P registers are identified with single letter abbreviations. All other named data registers are identified with a three character abbreviation not ending in F. The lowest order bit in a data register is always called bit zero. This bit is considered to be on the right end of the register. The bits are then arranged in order with the highest order bit on the extreme left end. The first bit in a data register is then the same as bit zero. The 18th bit in a data register is the same as bit 17. Both types of terminology are used to refer to individual data bits in this manual. All references to register bit position are decimal rather than octal.

Control condition names

Control conditions frequently become complex, and occasionally a group of control conditions appears repeatedly in a number of areas of the CPU. When this happens the group may be identified by a name to save repetitive listing of the members of the group. These names are multiple word designations which are exactly repeated in identifying the control condition wherever it is mentioned in this manual. All such group names are listed in the glossary at the end of this manual, together with the location in this section where the group name is defined. In no case is a control group name abbreviated into a three character designation.

All control conditions must be statically resolved in the 27.5 nanoseconds of one clock period. Any condition lasting longer than one clock period must appear as a control flag. It is therefore possible to reduce all control conditions to a logical expression involving only flag names and data register names.

Instruction stack

The CPU instruction stack consists of a number of parts as illustrated in figure 2-1 on the following page. The principal units in the instruction stack are the IAS and the IWS. A coincidence test is made between (P) and the content of each IAS rank during each clock period. Whenever a coincidence occurs, the corresponding 60 bit instruction word in the IWS is sent to the CIW register. If more than one coincidence occurs, all are sent simultaneously to the CIW register and the results are merged as a logical sum.

Instruction word stack (IWS)

The IWS is a group of twelve 60 bit registers which hold program instruction words for execution. The twelve registers are individually identified by rank. The rank one register contains the oldest data in the stack. If the IWS contains sequential program instruction words, the rank one register content will correspond with the lowest storage address in the instruction stack. The rank 12 register contains the last word to enter the stack. This register is loaded directly from SCM.

All registers in the IWS are clear/enter type registers with gated clock pulse control. That is, the information in each rank of the IWS will remain there indefinitely until a specific control condition gates a clock pulse to clear/enter the rank with new information. This condition is the "shift stack" condition. The information in each rank of the IWS moves through a delay path to the input of the next lower rank during each clock period. At the end of a clock period in which a "shift stack" condition exists, each rank is cleared and simultaneously entered with the information from the next highest order rank. The information in rank one is discarded. New information arriving from SCM is entered in rank 12.

Instruction address stack (IAS)

The IAS is a group of twelve 18 bit registers which hold relative SCM program addresses on a one-for-one basis with the program words in the IWS. The rank one register in the IAS contains the oldest address in the stack, and this address corresponds to the SCM location from which the word in rank one of the IWS was read. All ranks of the IAS are clear/entered with information from the next highest order rank on a "shift stack" condition in a manner analogous to that for the IWS. The address in rank one of the IAS is discarded. An address is read into rank 12 of the IAS from the NSA register. This address corresponds to the SCM address for the word arriving at rank 12 of the IWS.

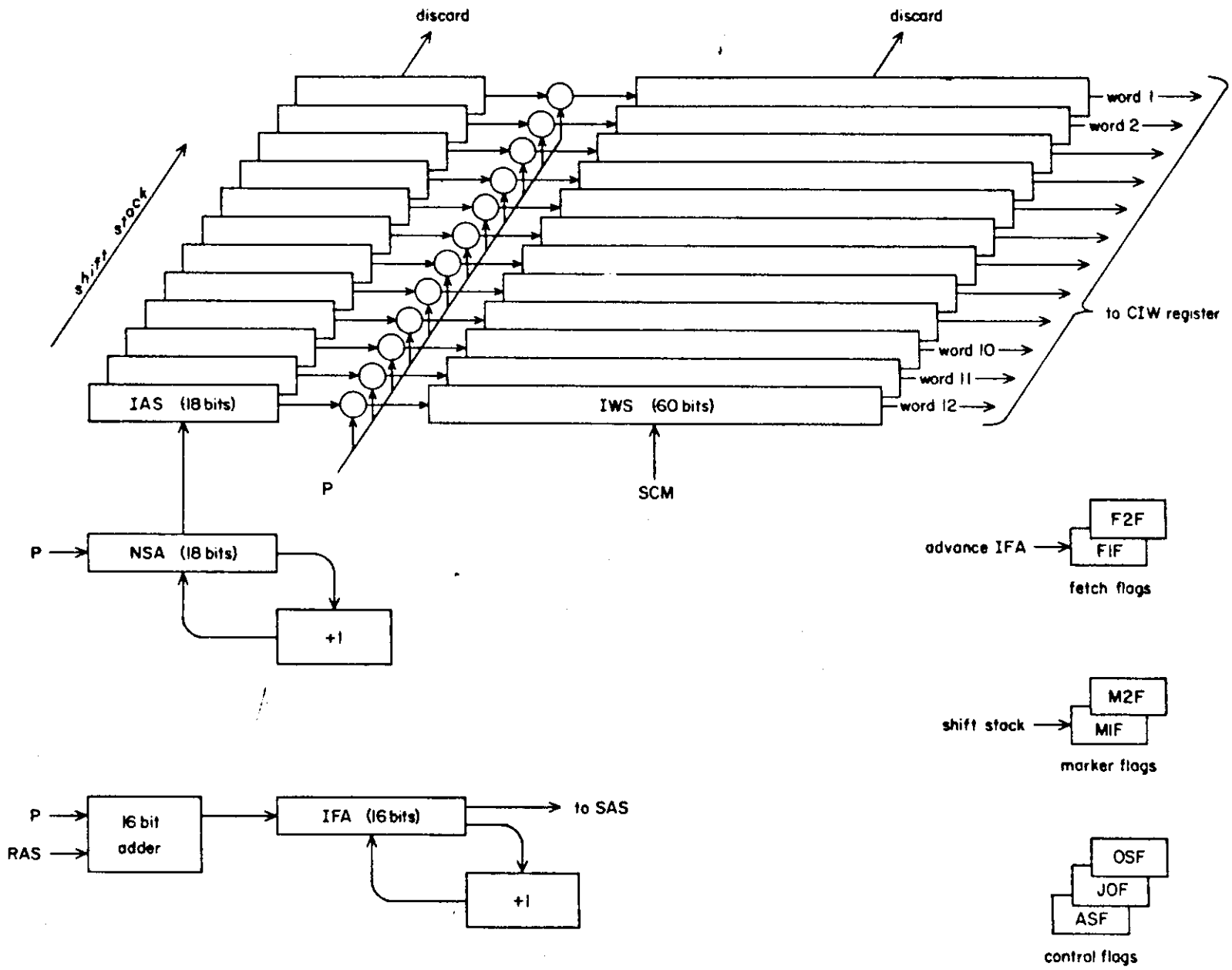


Fig. 2-1 CPU Instruction Stack

Next stack address register (NSA)

The NSA register is an 18 bit register which contains the SCM relative address for the next sequential word required by the IWS. This register is a clear/enter type register with gated clock pulse control. There are two possible sources for data entering this register. The content of the P register is entered in the NSA register at the end of a clock period in which a "P to NSA" condition exists. The content of the NSA register is increased by one count from the previous value at the end of a clock period in which the ASF is set. If both conditions exist in the same clock period the "P to NSA" condition takes priority.

If a "shift stack" condition exists during a clock period in which the ASF is not set, the address entered in the IAS rank 12 register is a copy of the content of the NSA register. If a "shift stack" condition exists during a clock period in which the ASF is set, the address entered in the IAS rank 12 register is one count greater than the value in the NSA register. In this second case the address entered in the IAS rank 12 register is generally the same as the new address being simultaneously entered in the NSA register. The only exception would be when a "P to NSA" condition and ASF occur in the same clock period.

Instruction fetch address register (IFA)

The IFA register is a 16 bit register which contains the next sequential absolute SCM address for fetch reference to the IWS. This register is a clear/enter type register with two possible sources of data. The sum of (P) + (RAS) is entered in the IFA register at the end of a clock period in which an "enter IFA" condition exists. This sum is formed in a static network treating both quantities as 18 bit positive integers and saving only the lowest order 16 bits. The IFA register content is increased one count from its previous value at the end of a clock period in which an "advance IFA" condition exists. If neither of these two conditions exists the IFA register is reset to its previous value at the end of the clock period. If both conditions exist in the same clock period the "enter IFA" condition takes priority.

The content of the IFA register is sent to the SAS for a SCM read reference when a program instruction fetch occurs. This reference is requested by the FIF and is described in the section on SCM.

Fetch one word flag (F1F)

The F1F is set whenever the instruction stack requires a word from SCM and the request has not yet been honored by the SAS. This flag is a clear/enter bit register which is forced clear on a dead start. The bit is set at the end of a clock period in which any one or more of the following conditions are satisfied.

(1) "Read stack" & "go issue" & no "coincidence" & OSF & no M1F & no JOF
(2) "Read stack" & "go issue" & no "coincidence" & JCF & no F1F
(3) "Read stack" & "go issue" & "word 12" & no M2F
(4) "Read stack" & "go issue" & "word 11" & no M1F
(5) F1F & no "advance IFA"
(6) F1F & F2F

Fetch two words flag (F2F)

The F2F is set whenever the instruction stack requires two words from SCM and neither request has yet been honored by the SAS. The F1F is always set when F2F is set. This flag is a clear/enter bit register which is forced clear on a dead start. The bit is set at the end of a clock period in which any one or more of the following conditions are satisfied.

(1) "Read stack" & "go issue" & no "coincidence" & OSF & no M1F & no JOF
(2) "Read stack" & "go issue" & no "coincidence" & JCF & no F1F
(3) "Read stack" & "go issue" & "word 12" & no M2F & F1F & no "advance IFA"
(4) "Read stack" & "go issue" & "word 12" & no M1F
(5) F2F & no "advance IFA"

One word marker flag (M1F)

The M1F is set whenever the instruction stack requires a word from SCM which has not yet arrived at the IWS. This flag is a clear/enter bit register which is forced clear on a dead start. The bit is set at the end of a clock period in which any one or more of the following conditions are satisfied.

(1) "Read stack" & "go issue" & "word 12"
(2) "Read stack" & "go issue" & "word 11" & no "shift stack"
(3) M1F & no "shift stack"
(4) "P to NSA"
(5) M1F & M2F

Two word marker flag (M2F)

The M2F is set whenever the instruction stack requires two words from SCM which have not yet arrived at the IWS. The M1F is always set when the M2F is set. This flag is a clear/enter bit register which is forced clear on a dead start. The bit is set at the end of a clock period in which any one or more of the following conditions are satisfied.

(1) "Read stack" & "go issue" & "word 12" & no "shift stack"
(2) M2F & no "shift stack"
(3) "P to NSA"

Jump out of stack flag (JOF)

The JOF is set when a branch instruction is executed which requires jumping to a program address not currently held in the IAS. This flag is a bit register with a separate set and a separate clear input path. The bit remains one from the time the flag is set until it is specifically cleared. The set and clear conditions can never exist in the same clock period.

Set condition: "read stack" & "go issue" & no "coincidence" & OSF & no M1F & no JOF
or: "read stack" & "go issue" & no "coincidence" & JCF & no F1F

Clear condition: "P to NSA"

Out of stack flag (OSF)

The OSF is set whenever the CIW register is ready for the next word from the IWS and no word is available. This flag is a bit register with a separate set and a separate clear input path which is forced clear on a dead start. The bit remains one from the time the flag is set until it is specifically cleared. The set and clear conditions can never exist in the same clock period.

Set condition: "read stack" & "go issue" & no "coincidence"

Clear condition: "read stack" & "go issue" & "coincidence"

Advance stack flag (ASF)

The ASF is a clear/enter bit register which is set at the end of a clock period in which a "shift stack" condition exists. The sole purpose of this flag is to advance the count in the NSA register in the clock period following a "shift stack" condition.

Set condition: "shift stack" & no "P to NSA"

"P to NSA" condition

This condition gates a clock pulse which causes the NSA register to clear and enter the contents of the P register. This condition also clears the JOF and sets both marker flags. This condition signals the arrival of the last fetch word requested prior to a program branch The NSA register is then set to the new program sequence address, and the marker flags are set in anticipation of the instruction words for the new program sequence.

Condition: JOF & no M1F
or: JOF & no M2F & "shift stack"

"Word 11" condition

This condition exists whenever the content of IAS rank 11 is the same as (P). This coincidence implies that the CIW register is about to read the next to last word in the IWS. When this word is read to the CIW register the F1F is set to request an instruction word from SCM.

"Word 12" condition

This condition exists whenever the content of IAS rank 12 is the same as (P). This coincidence implies that the CIW register is about to read the last word in the IWS. When this word is read to the CIW register the fetch flags are set so as to request a total of two instruction words from SCM.

"Coincidence" condition

This condition exists whenever an IAS rank contains the same address as the P register. This condition exists whenever the "word 11" condition or the "word 12" condition exists.

"Enter IFA" condition

This condition causes the IFA register to clear and enter the contents of the P register. This occurs on a program branch resulting in a jump out of stack. This condition exists when one or more of the three groups listed below are satisfied.

Condition: No "coincidence" & JCF & no F1F
or: No "coincidence" & OSF & no M1F & no JOF
or: RJF & no M1F

Other conditions

There are four conditions mentioned in this section which are not defined here because they originate in other parts of the CPU. The "shift stack" condition originates in the SCM destination control. This condition exists when a word is moving from SCM to the instruction stack. The "advance IFA" condition originates in the SCM access control. This condition exists when (IFA) has been accepted by the SAS as a result of a F1F request. The "read stack" and "go issue" conditions originate in instruction issue control. These two conditions taken together imply that the CIW register is ready for an instruction word from the IWS.

Straight line code

The program address is advanced sequentially in straight line code. The instruction words are requested for the stack two words ahead of the word currently being executed. As a result, coincidence between (P) and (IAS) occurs only on rank 11 and rank 12 registers. When a word is read from the IWS rank 11 register, the F1F is set to request another word. If the requested word arrives before program execution advances to the next word, the stack will have shifted data one rank and the next word will again be in rank 11. If the requested word does not arrive before program execution advances to the next word, a word will be read from rank 12 of the IWS. When this occurs a second word is requested from SCM to accelerate the stack filling process.

The chronological sequence of events in fetching a new instruction word is shown in the following timing chart. This chart assumes no delays in the SCM access control due to conflicts with other SCM references. If conflicts occur the arrival of the instruction word at the IWS will be delayed.

CP00 Rank 11 coincidence in the IAS.
Transmit IWS rank 11 word to the CIW register.
Set F1F.
Set M1F.

CP01 Transmit (IFA) to the SAS. Tag for read to IWS.
Advance (IFA).
Clear F1F.

Fetch address leaves SAS for a SCM bank.

CP02 SCM bank read/write cycle begins.

CP04

CP05

CP06 Instruction word reads to SCM bank operand register.

CP07 Transmit instruction word to the IWS.
Shift the IAS and the IWS one word position.
Transmit (NSA) to the IAS.
Set ASF.
Clear M1F.

CP08 Advance (NSA).
Clear ASF.

If program execution in the sequence listed above has progressed to the next instruction word before clock period 8, the next word for the CIW register will be read from IWS rank 12. When this happens, a second SCM request is initiated and another sequence of events similar to those listed above will overlap the above sequence. The amount of overlap will depend on the time the next instruction word is required.

It is possible for program execution to proceed so rapidly that the instruction word read from rank 12 of the IWS has been executed before the first of the two requested words has arrived. When this happens, the OSF is set and the program execution must wait for the arrival of the first instruction word.

Branch in stack

Program execution may reach a branch instruction and the destination address is already in the IAS. When this occurs the P register content is altered to the new program address. A coincidence occurs in the IAS during the following clock period, and the corresponding word is read from the IWS to the CIW register. The jump is then completed without a SCM reference for a new instruction word. When this situation occurs the branch instruction is executed in a total of three clock periods.

Timing charts for this case are listed in part three of this manual under the branch instruction description.

Branch out of stack

Program execution may reach a branch instruction and the destination address is not in the IAS. When this occurs the P register content is altered to the new program address. No coincidence occurs in the IAS during the following clock period, and the OSF and JOF are both set. A new address is entered in the IFA register and two words are requested from SCM to begin the new program sequence. When this situation occurs the branch instruction is executed in a total of 11 clock periods, assuming no conflicts delay the first instruction fetch in SCM.

Timing charts for this case are listed in part three of this manual under the branch instruction description.

It is possible that a branch out of stack occurs when the destination address corresponds to a program word which has already been requested from SCM as a result of sequential two word read ahead. If the word has not actually arrived at the IWS at the time of the branch test, the jump out of stack occurs and a duplicate of the first word in the new sequence is read from SCM. Execution of the new sequence can begin as soon as the earlier word arrives at the IWS.

Hole in the stack

It may happen that several small program sequences reside in the instruction stack at the same time. Program execution may branch back and forth between two such sequences. The execution of the sequence occupying the lower ranks of the instruction stack may branch in such a way as to continue sequential execution into a program area not loaded into the stack on the initial pass. When this happens it is possible for the next sequential instruction word to be missing in the stack and no request has been made for it because rank 11 or rank 12 were not involved.

This situation is equivalent to a branch out of stack with no branch instruction involved. The OSF is set as soon as the missing word is detected. The combination of OSF and no M1F sets the JOF and begins the jump out of stack sequence as if a branch instruction had been executed.

Duplicate entries in stack

It is possible for duplicate words to appear in the instruction stack. One way this may happen is on a jump out of stack with the first word of the new sequence already on the way to the IWS because of two word read ahead. Another way this may happen is on a jump backward in the program code and then sequential advancing into the area of program code already in the stack. No harm is done by duplicate words in the stack as long as the program code has not been altered in SCM between the two references. Coincidence occurs on both words simultaneously and both are sent to the CIW register. The data is merged in a logical sum network and the result will be the same as either individual word.

Instruction issue

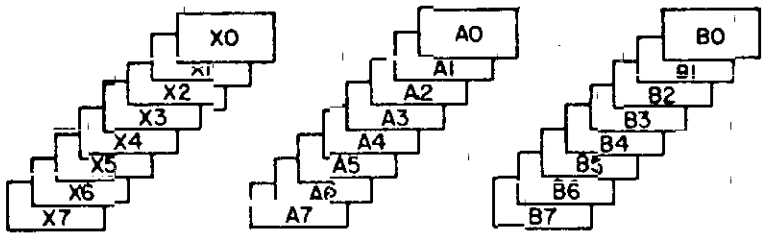
Program instruction words are read one at a time from the instruction stack into the current instruction word (CIW) register for execution. Each instruction word is divided into four 15 bit parcels as described in part one of this manual. There may be as many as four instructions in the CIW register at one time. These instructions must be executed in sequence and the proper allowance made for the mixture of one parcel and two parcel instruction formats.

An instruction "issues" from the CIW register when the conditions in the functional units and operating registers are such that the functions required in the instruction execution may be performed to completion without conflicting with a previously issued instruction. The issue process requires one clock period. During this clock period the g, h, i, j, and k designators in the instruction are sent to all appropriate parts of the CPU. The proper control flags are set at the end of this clock period to execute the functions required in the instruction. At the end of the clock period the data in the CIW register is altered to position the next instruction for execution. Once an instruction has issued from the CIW register it must be executed to completion in a fixed time framework. No delays are allowable from issue to delivery of data to the destination operating registers.

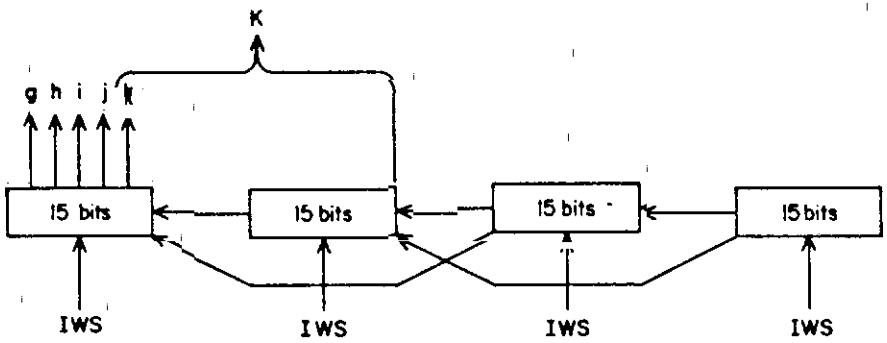
Current instruction word register (CIW)

The CIW register is a 60 bit register which is divided into four 15 bit parcels. All four parcels are loaded in one clock period when an instruction word is read from the IWS. The highest order parcel in the instruction word is issued first. The other parcels are then shifted left in the CIW register by either 15 bits or 30 bits, depending on the instruction format for the instruction issued. The lower order parcels are replaced with zeros as the data is shifted left in the register. This process is illustrated in figure 2-2 on the following page.

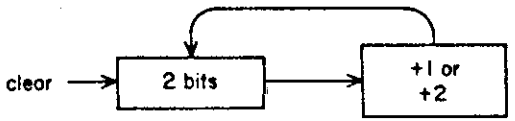
This register is a clear/enter type register with gated clock pulse control. There are three sources of data for each bit of the CIW register. These sources are the IWS, a one parcel shift path, and a two parcel shift path. The register is cleared and entered with new data only at the end of those clock periods in which conditions "go issue" and "registers free" exist. One of the three data sources is selected by the conditions listed below. If no data source is selected a 60 bit word of zeros is read into the CIW register.



Register reservation flags



CIW register



PKR register

Fig. 2-2 CPU Instruction Issue

Condition: "go issue" & "registers free" & "read stack"

This condition causes the selected contents of the IWS to enter the CIW register. If there is no coincidence between (P) and any rank of the IAS, zeros are read into the CIW register. If there is more than one rank of the IAS which coincides with (P), the selected words in the IWS are merged in a bit by bit logical sum network. The resulting 60 bit word is then entered in the CIW register.

Condition: "go issue" & "registers free" & "one parcel"

This condition causes the contents of the CIW register to shift left one parcel position. The instruction in the upper parcel position is discarded. The lowest order parcel position is filled with zeros.

Condition: "go issue" & "registers free" & "two parcel"

This condition causes the contents of the CIW register to shift left two parcel positions. The information in the upper two parcel positions is discarded. The lower two parcel positions are filled with zeros.

Parcel counter register (PKR)

The PKR register is a two bit clear/enter type register with gated clock pulse control. The content of this register indicates which parcel in the original instruction word is currently positioned in the CIW register for execution. A zero quantity in the PKR register indicates that the original word is in the CIW register and no shifts have taken place. (PKR) = 1 implies that the original instruction word has been shifted one parcel position in the CIW register prior to this clock period. The second parcel is then positioned for execution. (PKR) = 2 implies a two parcel shift. (PKR) = 3 implies a three parcel shift.

The PKR register is cleared and entered with a new value whenever the conditions "go issue" and "registers free" exist. The value entered is a function of the conditions listed below.

Set (PKR) to 1: "one parcel" & (PKR) = 0

Set (PKR) to 2: "one parcel" & (PKR) = 1
or: "two parcel" & (PKR) = 0

Set (PKR) to 3: "one parcel" & (PKR) = 2
or: "two parcel" & (PKR) = 1

X reservation flags

There are eight control flags associated with the X registers. These flags are bit registers with separate set and separate clear inputs which are forced clear on a dead start. The bit remains one from the time the flag is set until it is specifically cleared. The set and clear conditions can never exist in the same clock period. A flag is set to reserve a specific X register when an instruction issues from the CIW register which will deliver a result to that register. The flag is cleared when the result is actually transmitted to the X register. This flag prevents subsequent instructions from reading the contents of this X register until the new data has been delivered.

One of the X register reservation flags is set at the end of a clock period in which the condition "go issue" & "registers free" & "set Xi reservation" exists. The X register reservation flag is specified by the i designator in the CIW register.

One of the X register reservation flags is set at the end of a clock period in which the condition "go issue" & "registers free" & "set Xj reservation" exists. The X register reservation flag is specified by the j designator in the CIW register.

One of the X register reservation flags is cleared at the end of a clock period in which a "clear Xd reservation" condition exists. This condition originates in the register access control and indicates that the designated X register is receiving data from a functional unit. Only one X register can receive data in a given clock period.

A reservation flags

There are eight control flags associated with the A registers. These flags are bit registers with separate set and separate clear inputs like the X reservation flags. The A reservation flags perform the same function for the A registers as the X reservation flags perform for the X registers. There is a "set Ai reservation" condition and a "clear Af reservation" condition which perform functions equivalent to the corresponding conditions for the X registers. There is no condition "set Aj reservation."

B reservation flags

There are eight control flags associated with the B registers. These flags are bit registers with separate set and separate clear inputs like the X register flags. The B reservation flags perform the same function for the B registers as the X reservation flags perform for the X registers. There is a "set Bi reservation" condition, a "set Bj reservation" condition, and a "clear Be reservation" condition which perform equivalent functions to the corresponding X register conditions.

"Go issue" condition

This condition indicates that the functional units, register destination paths, and storage access path are such as to allow issue of the current instruction in the CIW register. This condition does not take into account the possible register reservation problems. This condition is defined as follows:

```
Condition:  NBF & g = 4 & h = 4,5 & no divide busy flag
      or:   NBF & g = 3 & h = 0,1,2,3,4,5 & no register #4 mark bit &
            no go multiply flag
      or:   NBF & g = 4 & h = 0,1,2 & no "read to X reference" & no
            divide time 15 & no go multiply flag
      or:   NBF & g = 0 & h = 1 & i = 6,7
      or:   NBF & g = 0 & h = 1 & i = 4,5 & no LCM busy flag
      or:   NBF & g = 2 & h = 4,5 & no register #3 mark bit & no go
            add flag
      or:   NBF & g = 0 & h = 0
      or:   NBF & g = 6 & no go normalize flag & no go read channel flag
      or:   NBF & g = 5
      or:   NBF & g = 4 & h = 3,7 & no register #2 mark bit & no go
            normalize flag
      or:   NBF & g = 3 & h = 6,7 & no register #2 mark bit & no go
            normalize flag
      or:   NBF & g = 2 & h = 0,1,2,3,6,7 & no register #2 mark bit &
            no go read channel flag & no go normalize flag
      or:   NBF & g = 1,7 & no register #2 mark bit & no go normalize flag
      or:   GJF & "fall through"
      or:   g = 4 & h = 6
      or:   PXF
      or:   JCF
      or:   "block copy completed"
```

"Registers free" condition

This condition indicates that the operating registers involved in the instruction waiting to issue are now free. That is, there are no register reservation flags set for an A, B, or X register that is either an operand or a destination register for the current instruction in the CIW register. This condition is defined as follows:

```
not:  Xi reservation flag & g = 1,2,3,7
nor:  Xi reservation flag & g = 4 & h = 0,1,2,3,4,5,7
nor:  Xi reservation flag & g = 5 & i = 1,2,3,4,5,6,7
nor:  Xj reservation flag & g = 0 & h = 1 & i = 4,5
nor:  Xj reservation flag & g = 0 & h = 3
nor:  Xj reservation flag & g = 1 & h = 0,1,2,3,5,6,7
nor:  Xj reservation flag & g = 3
nor:  Xj reservation flag & g = 4 & h = 0,1,2,4,5
nor:  Xj reservation flag & g = 5,6,7 & h = 2,3
nor:  Xk reservation flag & g = 0 & h = 1 & i = 4,5
nor:  Xk reservation flag & g = 1 & h = 1,2,3,4,5,6,7
nor:  Xk reservation flag & g = 2 & h = 2,3,4,5,6,7
nor:  Xk reservation flag & g = 3
nor:  Xk reservation flag & g = 4 & h = 0,1,2,4,5,7
nor:  Bi reservation flag & g = 0 & h = 2
nor:  Bi reservation flag & g = 0 & h = 4,5,6,7
nor:  Bi reservation flag & g = 6
nor:  Bj reservation flag & g = 0 & h = 1 & i = 3,6,7
nor:  Bj reservation flag & g = 0 & h = 4,5,6,7
nor:  Bj reservation flag & g = 2 & h = 2,3,4,5,6,7
nor:  Bj reservation flag & g = 5,6,7 & h = 1,6,7
nor:  Bk reservation flag & g = 5,6,7 & h = 3,4,5,6,7
nor:  Bk reservation flag & g = 0 & h = 1 & i = 6,7
nor:  Ai reservation flag & g = 5
nor:  Aj reservation flag & g = 5,6,7 & h = 0,4,5
```

"Two parcel" condition

This condition indicates that the instruction currently positioned in the upper parcel of the CIW register is a two parcel instruction and there are more instructions in the word. When the current instruction issues this condition causes the data in the CIW register to shift left two parcel positions. This condition is defined as follows:

```
Condition:  no JCF & (PKR) = 0,1 & g = 0 & h = 2,3,4,5,6,7
      or:   no JCF & (PKR) = 0,1 & g = 0 & h = 1 & i = 0,1,2
      or:   no JCF & (PKR) = 0,1 & g = 5,6,7 & h = 0,1,2
```

"One parcel" condition

This condition indicates that the instruction currently positioned in the upper parcel of the CIW register is a one parcel instruction and there are more instructions in the word. When the current instruction issues this condition causes the data in the CIW register to shift left one parcel position. This condition is defined as follows:

```
not: JCF
nor: (PKR) = 3
nor: g = 0 & h = 0
nor: g = 0 & h = 1 & i = 0,1,2,3
nor: g = 5,6,7 & h = 0,1,2
nor: g = 0 & h = 2,3,4,5,6,7
```

"Read stack" condition

This condition indicates that the instruction currently positioned in the upper parcel of the CIW register is the last instruction in the word. When this instruction issues the CIW register will be loaded with a new 60 bit word from the IWS. This condition is defined as follows:

```
not: "dead start"
nor: JOF
nor: RIF
nor: XSF
nor: g = 0 & h = 0 & no JCF & no OSF
nor: "one parcel"
nor: "two parcel"
```

"Set Xi reservation" condition

This condition together with "go issue" & "registers free" causes the Xi reservation flag to set. This condition is defined as follows:

```
not: g = 0
nor: g = 6
nor: g = 4 & h = 6
nor: g = 5 & i = 0,6,7
```

"Set Xj reservation" condition

This condition together with "go issue" & "registers free" causes the Xj reservation flag to set. This condition is defined as follows:

Condition: g = 0 & h = 1 & i = 4

"Set Ai reservation" condition

This condition together with "go issue" & "registers free" causes the Ai reservation flag to set. This condition is defined as follows:

Condition: g = 5

"Set Bi reservation" condition

This condition together with "go issue" & "registers free" causes the Bi reservation flag to set. This condition is defined as follows:

Condition: g = 6 & i = 1,2,3,4,5,6,7

"Set Bj reservation" condition

This condition together with "go issue" & "registers free" causes the Bj reservation flag to set. This condition is defined as follows:

Condition: g = 2 & h = 4,5,6 & j = 1,2,3,4,5,6,7
or: g = 0 & h = 1 & i = 6,7 & j = 1,2,3,4,5,6,7

Boolean unit

The boolean unit executes those CPU instructions which require bit by bit data manipulation. This includes the logical operations for instructions 11, 12, 13, 15, 16, and 17 plus the transmissive operations for instructions 10, 14, 26, and 27. Data transmission paths to, and from, the boolean unit are illustrated in figure 2-3 on the following page.

There are three data input registers for the boolean unit. These are clear/enter type registers which are cleared and entered with new data at the end of each clock period. The contents of the Bj, Xj, and Xk registers are transmitted to the boolean unit each clock period without regard to the instruction in the CIW register. These operands are then available in the boolean unit in the following clock period.

There are several bits of control information stored in the boolean unit at the end of each clock period. The g and h designators are sent to the boolean unit from the CIW register in much the same manner as the data to the data input registers. In addition the go boolean flag is set only at the end of a clock period in which an instruction requiring the boolean unit issues from the CIW register. The go boolean flag is therefore set during the clock period following issue of a boolean instruction.

Data in the boolean unit input registers is merged in a static network for transmission to the destination registers. The type of logical operation and selection of data paths in this static network is determined by the control flags with the delayed g and h designator values. The data is actually transmitted to the destination registers only during a clock period when the go boolean flag is set. If the go boolean flag is set during a given clock period, a boolean instruction must have issued during the previous clock period. The data in the input registers must therefore correspond with the data described by the j and k designators in that instruction. The g and h values stored in the boolean unit must also describe the mode of operation for the instruction. The go boolean flag is then a necessary and sufficient condition for transmitting the output of the static network to the destination registers.

Data from the several functional units is merged into the operating registers with one common data path to the X registers, one to the A registers, and one to the B registers. It is important that only one functional unit transmit data on an input path during any given clock period. This is controlled by the conditions for issue and is not a concern of the individual functional units.

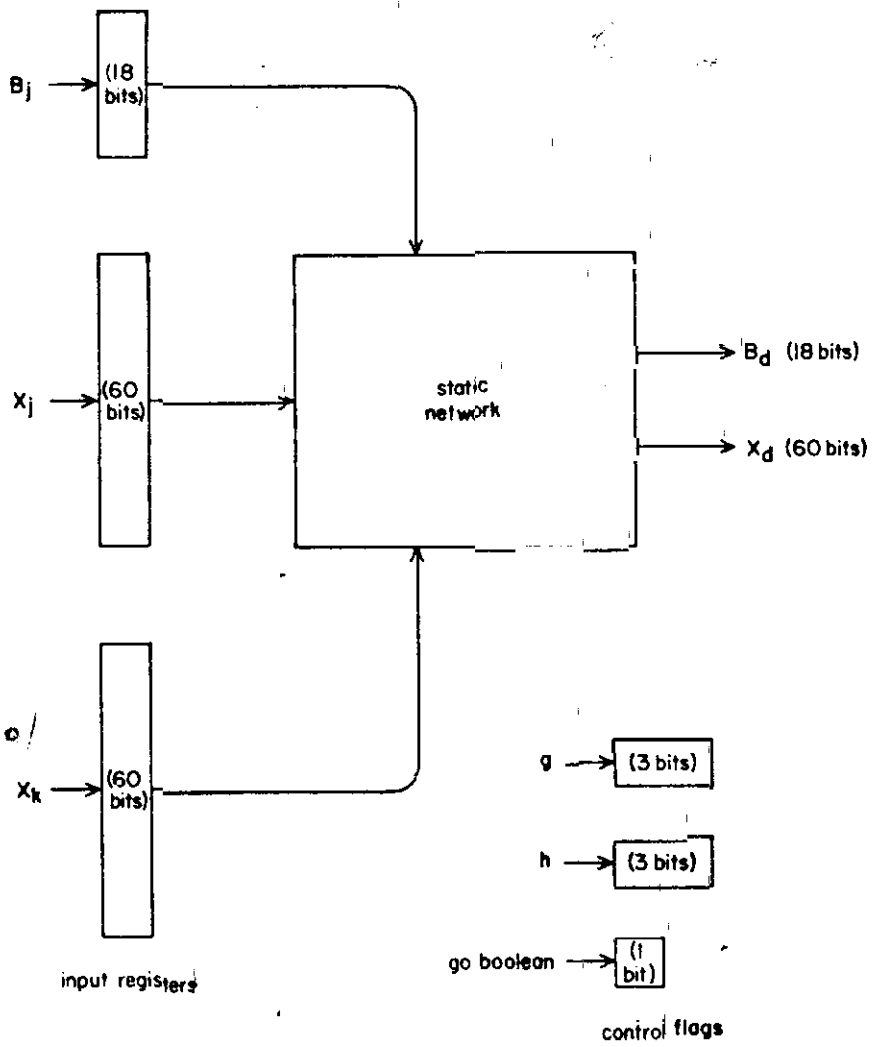


Fig. 2-3 Boolean Unit

Go boolean flag

The go boolean flag is set in the boolean unit at the end of a clock period in which an instruction requiring the boolean unit issues from the CIW register. Data is transmitted to the boolean unit input registers concurrent with the instruction issue. The go boolean flag then gates the data to the destination register. Two clock periods are therefore required to execute a boolean type instruction. During one clock period data moves from the operating registers to the boolean unit input registers. During the following clock period data moves from the boolean unit input registers through the static selection network and back to the operating registers.

Conditions for setting the go boolean flag are as follows:

Condition: "go issue" & "registers free" & g = 1
or: "go issue" & "registers free" & g = 2 & h = 6,7

Timing charts for the individual instructions involving the boolean unit are listed in part three of this manual. Each such instruction is executed in two clock periods. The boolean unit is free to begin executing a new instruction every clock period. If a boolean type instruction does not issue in a given clock period the data in the boolean unit input registers is simply discarded in the following clock period. A typical timing sequence is listed below for the 15 instruction.

15 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the boolean unit.
Transmit (Xk) to the boolean unit.
Set Xi reservation flag.
Set go boolean flag.

Transmit data from the boolean unit to the Xi register.
Clear Xd reservation flag.
Clear go boolean flag.

Shift unit

The shift unit executes those CPU instructions which require shifting the entire 60 bit field of data within the operand word. This is required on instructions 20, 21, 22, 23, and 43. Organization and data paths within the shift unit are illustrated in figure 2-4 on the following page.

All instructions performed in the shift unit require two clock periods for execution. Data moves from the operating registers to the shift unit in the same clock period in which a shift instruction issues from the CIW register. Data moves from the shift unit back to the operating registers during the following clock period. A new instruction may be issued for execution in the shift unit each clock period.

Operands for the shift unit are of two types. A 60 bit word may be read from either the Xi register or the Xk register, depending on the type of instruction. A second operand is read from the Bj register, or from the CIW register, to determine the amount of shift for the 60 bit word. Those instructions reading the shift count from the CIW register treat the j designator and the k designator as a single six bit shift count.

The shift unit contains a single level of registers to hold the data and the control information at the end of a clock period in which a shift instruction issues from the CIW register. These registers are all of the clear/enter type, and are cleared at the end of every clock period. The principal input register holds 60 bits of data for the destination X register. This data passes through two levels of static networks before reaching the input register. The first network makes the selection between Xi or Xk on the basis of the instruction code in the CIW register. The Xk path is selected if the h designator in the CIW register has a value of 2, 3, 6, or 7. The Xi path is selected if the h designator has a value of 0, 1, 4, or 5.

The second level of static selection for the input register has three possible modes. One mode is used for the 20 series instructions which require a left shift of data. Another mode is used for the 20 series instructions which require a right shift of data. The third mode is used for the 43 instruction, which requires neither Xk nor Xi for an operand. In this last mode no data is selected for the input register, and all zeros are entered for the operand.

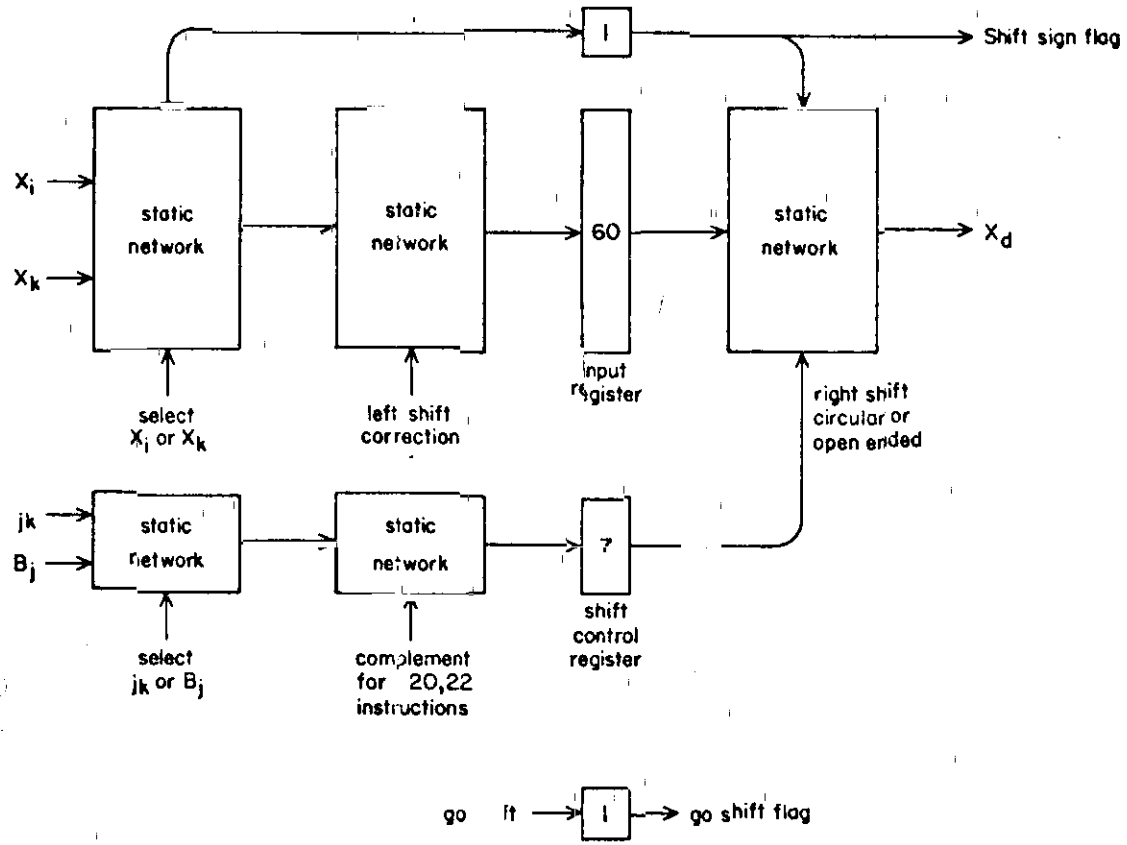


Fig. 2-4 Shift Unit

The shift unit has the provision for shifting a 60 bit field of data circularly to the right by any arbitrary number of bit positions. It also may shift the 60 bit field open ended to the right by blocking the flow of data around the end of the data field. In this later case zero bits are entered in the leftmost positions in the field as the data is shifted. There is no provision for shifting data to the left in the shift unit.

A left shift of data is simulated by a correction shift in the static network in front of the input register, and then a right circular shift between the input register and the destination X register. The static network between the input register and the destination X register is able to shift the data right by any value in a six bit shift count held in the shift control register. The static network shifts right in a circular mode, or in an open ended mode, depending on a seventh bit in the shift control register.

The shift control information held in the shift unit passes through two static networks before reaching the shift control register. The first static network selects either Bj or the CIW register as the source of the shift count. The Bj register path is selected for instructions 22 and 23. The jk portion of the CIW register is selected for instructions 20, 21, and 43. The second static network then complements the shift count for instructions 20 and 22. The resulting six bit quantity from the jk portion of the CIW register, or from the lowest order six bits of the Bj register, is entered in the shift control register. The seventh bit in the shift control register is set for a circular mode. This bit is set on the following condition:

```
condition:  g = 2 & h = 0
      or:   g = 2 & h = 2 & (Bj) positive
      or:   g = 2 & h = 3 & (Bj) negative
```

A special test is made to determine if (Bj) has more than six bits of significance. The lowest order 12 bits of (Bj) plus the sign bit of (Bj) are included in this test. The other five bits of (Bj) are ignored. If the Bj path is selected, and if the shift is open ended, the presence of higher order bits in (Bj) will cause the operand to shift off the end of the destination field and deliver all sign bits as a result. If the mode of the shift is circular, the lowest order six bits determine the shift count modulo 64, and the higher order bits are ignored.

Go shift flag

The go shift flag is set in the shift unit at the end of a clock period in which a shift instruction issues from the CIW register. This flag then gates the data from the static network in the shift unit to the destination X register input path. If the go shift flag is not set in a given clock period, the data from the shift unit in that clock period is discarded. The condition for setting the go shift flag is as follows:

condition: "go issue" & "registers free" & g = 4 & h = 3
or: "go issue" & "registers free" & g = 2 & h = 0,1,2,3

Shift sign flag

The shift sign flag is set when a negative operand is processed by the shift unit. The static network between the input register and the destination X register shifts data in a positive mode only. A negative operand is complemented at the front of this network before the shift is performed. The result is then complemented again at the input to the destination X register. The shift sign flag controls both of these operations. The condition for setting this flag is as follows:

condition: g = 2,3,6,7 & h = 2,3,6,7 & (Xj) negative
or: g = 2,3,6,7 & h = 0,1,4,5 & (Xi) negative
or: g = 0,1,4,5

Left shift mode

A left circular shift is simulated in the shift unit whenever the circular mode bit is set in the shift control register. A left shift correction occurs in the static network feeding the input register. This correction causes the 60 bit data word to shift left circularly by three bit positions. The static network between the input register and the destination X register shifts the data word right circularly by the shift count in the shift control register. This shift count is the modulo 63 negative of the desired left shift count. Since the field length is 60 the desired shift is simulated in a somewhat indirect manner as shown in the expression below:

3 - (63 - n) = n - 60 = n, where n is the desired shift count

Right shift mode

A right open ended shift is performed in the shift unit whenever the circular mode bit in the shift control register is not set. In this mode the left shift correction in the static network feeding the input register is omitted. This network transmits the unaltered operand into the input register. The static network between the input register and the destination X register performs a right open ended shift by the amount of the shift count in the shift control register.

Form mask mode

The shift unit operates in a special mode for executing the 43 instruction. In this mode the static network feeding the input register blocks the operand and enters all zeros in the input register. The shift sign flag is always set for this instruction. The circular mode bit in the shift control register is never set for this instruction. The static network between the input register and the destination X register then complements a field of all zeros, shifts the resulting field of all ones open ended to the right, and delivers the result to the X register input path. The data is complemented again at the input to the X register because the shift sign flag is set. The result is a variable field of ones in the upper portion of the word as controlled by the j and k designators in the 43 instruction.

Timing charts for instructions executed in the shift unit are listed individually for each instruction in part three of this manual. A typical timing sequence is listed below for the 20 instruction.

CP00 20 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit (Xi) to the shift unit.
Set Xi reservation flag.
Set go shift flag.

CP01 Transmit data from the shift unit to the Xd register.
Clear Xd reservation flag.
Clear go shift flag.

Normalize unit

The normalize unit executes CPU instructions 24 and 25. These two instructions are identical except that instruction 25 adds a round bit to the operand coefficient which is not present on instruction 24. Organization of the normalize unit is illustrated in figure 2-5 on the following page.

The normalize instructions require three clock periods for execution. Data moves from the operating registers to the normalize unit in the same clock period in which the instruction issues from the CIW register. At the end of this clock period all of the information necessary to complete the instruction is captured in a set of input registers. During the next clock period the coefficient and exponent portions of the floating point operand move through separate static networks to a set of normalize unit output registers. During the third clock period the data moves from the output registers through static networks to the destination operating registers. The control information flows through the normalize unit along with the operand data. This allows a new normalize instruction to issue every clock period for execution in the same normalize unit.

All registers in the normalize unit are of the clear/enter type and are cleared at the end of each clock period. The data and the control information flows continuously through the unit. When a normalize instruction issues from the CIW register the go normalize flag is set in the input register area. This flag is copied into a corresponding flag in the output register in the following clock period. When this later flag is set the data in the normalize unit output registers is sent to the X register and B register input data paths. When this flag is not set the data in the normalize unit output registers is discarded.

The content of register Xk is entered in the normalize unit input register at the end of every clock period. The exponent portion of the word is treated in one part of the normalize unit and the coefficient portion in another part. Both portions of the operand are complemented in a static network immediately following the input register if the original operand was negative. The remainder of the normalize operation is then performed on the resulting positive operand value. A static network in the coefficient part of the normalize unit determines the shift count required to normalize the coefficient. This is the same as the number of leading zeros in the coefficient at this point in the process. The shift count detector delivers a six bit shift quantity to the output register for use in normalizing the coefficient in the third clock period of the sequence. The shift count detector also delivers the same six bit quantity to the exponent part of the normalize unit for exponent correction.

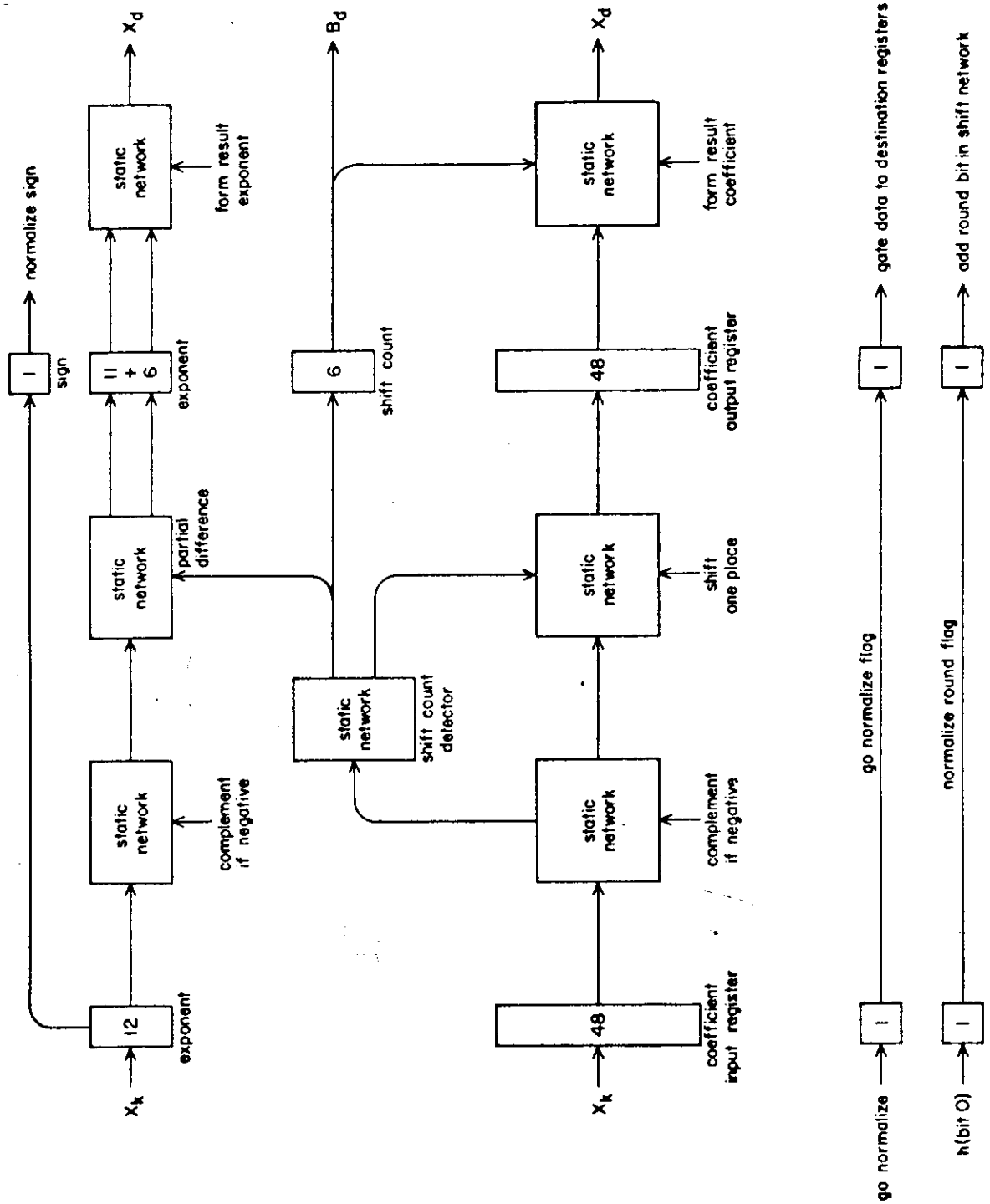


Fig. 2-5 Normalize Unit

One bit position of coefficient shift is executed between the coefficient input register and the coefficient output register. If the shift count from the shift count detector is odd a shift of one bit position in the coefficient is performed in this static network. If the shift count is even no shift occurs in this network. The remaining five bits of the shift count are interpreted and executed by the static network between the coefficient output register and the destination X register.

The exponent portion of the operand is complemented if the original word was negative. The highest order bit of the packed representation for the exponent is independently complemented to remove the bias and form an 11 bit ones complement value for the exponent. This quantity then enters a static network along with the six bit shift count from the shift count detector. This network forms a partial difference in an 11 bit ones complement mode. The 11 partial difference bits and the six possible borrow bits are held in the output register. The subtraction of the six bit shift count from the 11 bit operand exponent is then completed in a static network between the output register and the destination X register. The highest order bit of the exponent field is recomplemented in this network to restore the bias for packed floating point representation.

The sign of the original operand is held in a separate control flag in the normalize output register area. This flag controls the complement of the result at the input to the destination X register. If the original operand was negative this flag will be set and the sign of the result will be corrected at the input to the X register.

A test for special case operand is made on the exponent after the exponent is corrected for sign. If the operand is an overflow quantity or an indefinite quantity the shift count from the shift count detector is blocked. The operand then passes through the normalize unit unaltered. The quantity delivered to the destination B register in this case is zero.

If the coefficient portion of the operand is all zeros after the correction for operand sign, the shift count detector indicates a shift of 48 decimal bit positions. If this situation occurs in execution of a 25 instruction the round bit results in a normalized coefficient. If this situation occurs in execution of a 24 instruction the shift count of 48 decimal is treated as a special case, and the result delivered to the destination X register input path is all zeros. The shift count delivered to the destination B register is 48 decimal in either of these cases.

The subtraction of the six bit shift count from the operand exponent may result in an underflow of the floating point exponent range. This situation is detected, and the outputs from the static networks to the destination X register are blocked. In this case all zeros are sent to the X register data input path. The shift count delivered to the destination B register is not affected by this situation.

Go normalize flag

The go normalize flag is set at the end of every clock period in which a normalize instruction issues from the CIW register. This flag is copied during the following clock period to another flag in the normalize unit output register area. This flag in turn gates the data to the destination operating registers. If this flag is not set the data in the normalize unit output registers is discarded. The condition for setting the go normalize flag is as follows:

condition: "go issue" & "registers free" & g = 2 & h = 4,5

Normalize round flag

The normalize round flag is set at the end of every clock period in which a round normalize instruction issues from the CIW register. This flag has no significance unless the go normalize flag is also set. As a result, the normalize round flag can be set from a single bit of the h designator in the CIW register. This flag distinguishes between the 24 instruction and the 25 instruction in the operation of the normalize unit. When the normalize round flag is set, a 49th bit is added to the coefficient for the operand. The condition for setting the normalize round flag is as follows:

condition: h = 1,3,5,7

The timing charts for the 24 instruction and 25 instruction are essentially the same. These are listed in part three of this manual.

Long add unit

The long add unit executes CPU instructions 36 and 37. These two instructions involve 60 bit integer addition of two operands to form a 60 bit sum. The two instructions are executed in the same manner except that one operand is complemented prior to the addition in executing the 37 instruction. The organization of the long add unit is illustrated in figure 2 - 6 on the following page.

The long add instructions require two clock periods for execution. Data moves from the operating registers to the long add unit in the same clock period in which the instruction issues from the CIW register. Data moves from the long add unit back to the operating registers during the following clock period. A new instruction may be issued for execution in the long add unit each clock period.

The long add unit contains a single level of registers to hold the data and control information at the end of a clock period in which a long add instruction issues from the CIW register. These registers are all of the clear/enter type, and are cleared at the end of every clock period.

The contents of the Xj register and the contents of the Xk register are sent to the long add unit during each clock period. The quantity (Xk) is complemented in a static network if the current instruction in the CIW register is a 37 instruction. Specifically, the network complements (Xk) if the h designator has the value 1, 3, 5, or 7. The output of this static network and (Xj) are then merged in a second static network to form the partial sum of these two quantities in a 60 bit ones complement mode. The 60 partial sum bits and 60 possible carry bits from this second static network are then held in the long add unit registers for use in the following clock period.

During the second clock period in the execution of a long add instruction the 60 partial sum bits and 60 carry bits are merged in a static network to complete the addition operation in a 60 bit ones complement mode. The output of this static network then goes to the data input path to the X registers.

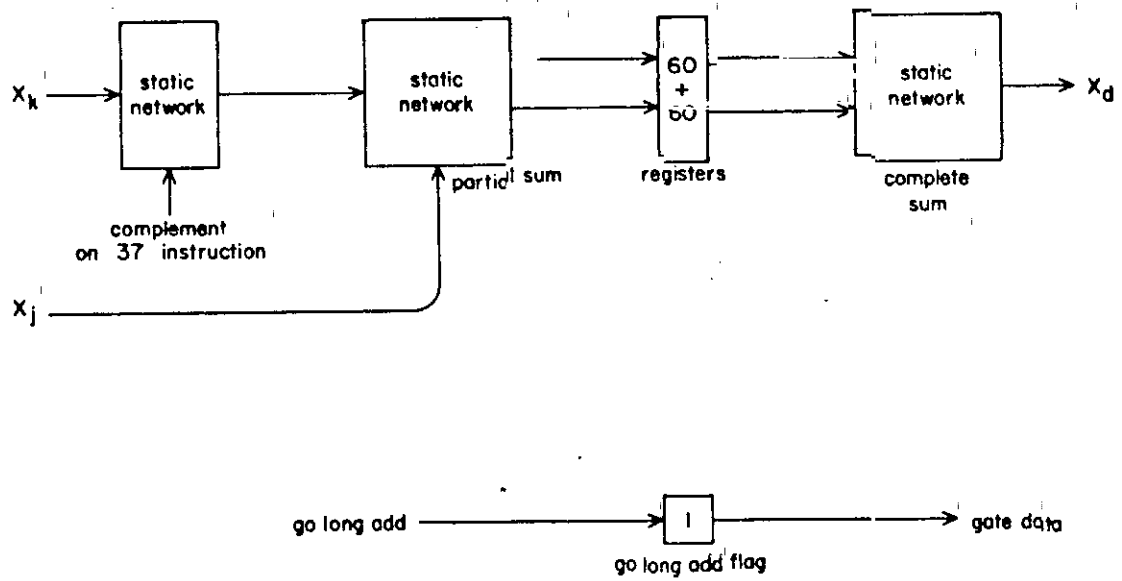


Fig. 2-6 Long Add Unit

Go long add flag

The go long add flag is set in the long add unit at the end of a clock period in which a long add instruction issues from the CIW register. This flag then gates the data from the static network in the long add unit to the destination X register input path during the following clock period. If the go long add flag is not set during a given clock period, the data in the long add unit is discarded. The condition for setting the go long add flag is as follows:

Condition: "go issue" & "registers free" & g = 3 & h = 6,7

Timing charts for instructions executed in the long add unit are listed separately for each instruction in part three of this manual. The timing is essentially the same for the 36 and 37 instructions and is listed below for the 36 instruction.

CP00 36 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit (Xj) to the long add unit.
Transmit (Xk) to the long add unit.
Set Xi reservation flag.
Set go long add flag.

CP01 Transmit data from the long add unit to the Xd register.
Clear Xd reservation flag.
Clear go long add flag.

Floating add unit

The floating add unit executes those CPU instructions which require addition of operands in floating point format. This requirement exists for instructions 30, 31, 32, 33, 34, and 35. The organization of the floating add unit is illustrated in figure 2-7 on the following page.

All instructions performed in the floating add unit require four clock periods for execution. Data moves from the operating registers to the floating add unit input registers in the same clock period in which a floating add instruction issues from the CIW register. Data moves from the input registers to intermediate registers in the second clock period of instruction execution. Data moves from the intermediate registers to the floating add unit output registers during the third clock period. Data moves from the output registers back to the destination X register in the last clock period.

All registers in the floating add unit are of the clear/enter type and are cleared at the end of every clock period. The data and the associated control information flow continuously through the unit. When a floating add instruction issues from the CIW register the go floating add flag is set in the floating add unit. This flag corresponds to data entering the input registers. The go floating add flag is copied through the following two clock periods as the data moves through the floating add unit. In the last clock period of execution the presence of this flag gates the data from the floating add unit output registers to the X register data input path. When this flag is not set the data in the floating add unit output registers is discarded.

A floating add instruction may issue from the CIW register each clock period for execution in the same floating add unit. The contents of the Xj register and the Xk register are stored in the floating add unit input registers at the end of every clock period. Two bits of the h designator are also stored in the input register area. These two bits of control information determine the mode of operation of the floating add unit as the data is processed.

The lowest order bit of the h designator in the CIW register determines whether the floating add unit performs an addition, or a subtraction, operation. If h = 1,3,5,7 a static network in the floating add unit complements (Xk) on the way to the input register area. The remainder of the floating add unit operation is always in an addition mode.

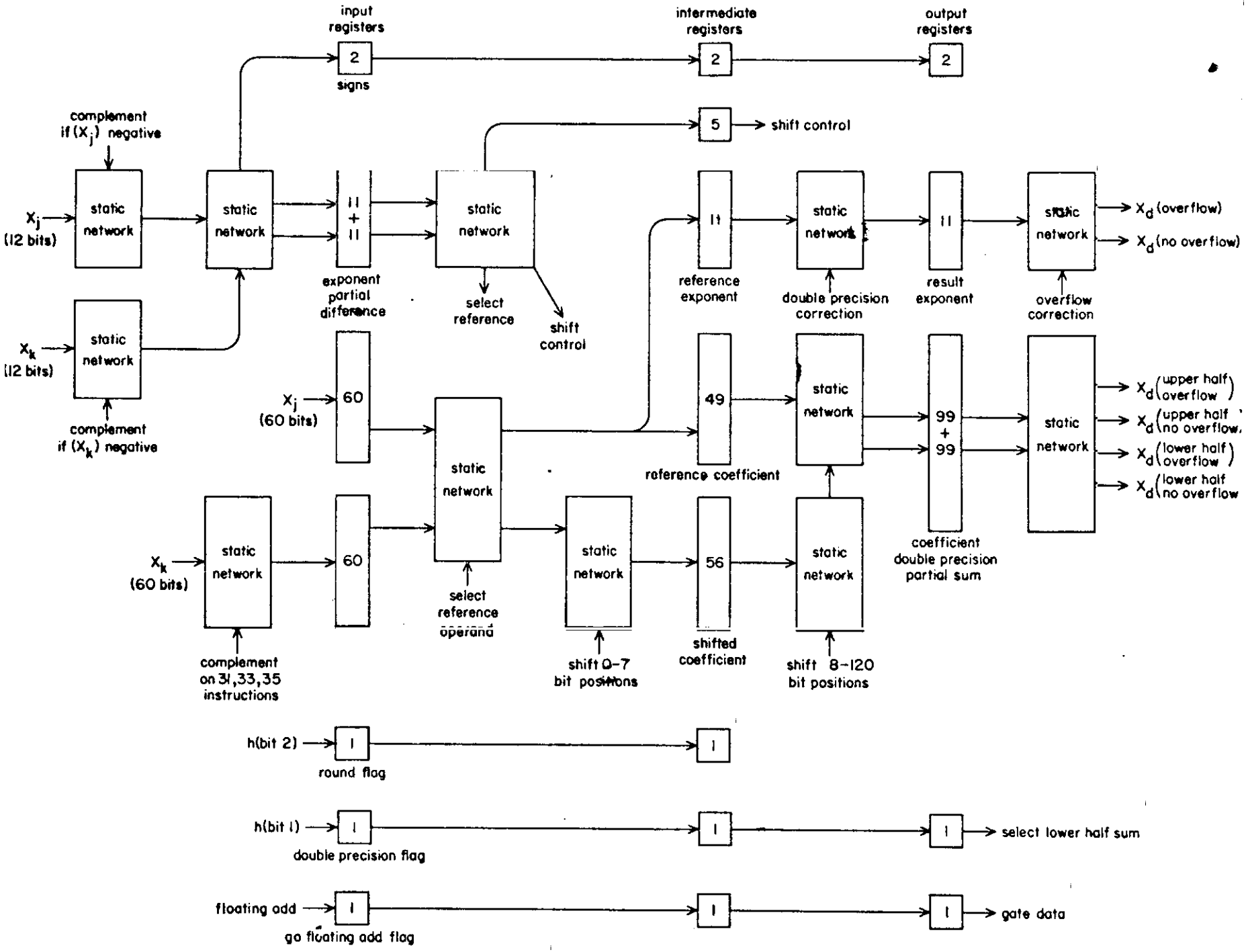


Fig. 2-7 Floating Add Unit

The upper 12 bits of (Xj) and (Xk) are transmitted to the floating add unit simultaneously over two data paths. One set of these exponent values is processed along with the coefficient values in complete 60 bit words. The other set of exponent values is treated in a special portion of the floating add unit which determines the difference of the two exponents. The upper 12 bits of (Xj) and the upper 12 bits of (Xk) each pass through a static network on the way to the input area in this special portion of the floating add unit. These static networks sense the sign of each operand and complement the exponent field if the word was negative. In addition, these networks complement the highest order bit in the exponent field to remove the bias associated with packed floating point representation of the numbers. The resulting two 11 bit exponents then merge in a static network which forms the partial difference in a 12 bit ones complement mode.

The partial difference of the exponent values is held in the input register for use during the following clock period. During this clock period a static network completes the subtraction process and determines the sign of the exponent difference. The Xj exponent is subtracted from the Xk exponent. If the sign of the difference is negative, the Xj quantity is selected as the reference operand and the Xk quantity is selected as the operand to be shifted. If the sign of the exponent difference is positive, the Xk quantity is selected as the reference operand and the Xj quantity is selected as the operand to be shifted. This same static network determines the amount of shift required to align the coefficients for addition. If the exponent difference is negative its value is complemented. The value of this resulting integer is the number of bit positions of shift required to align the coefficients.

The two 60 bit operands are held in the input register area while the exponent difference is determined. During the second clock period of execution a selection is made between the two operands. One exponent is selected as the reference exponent and is stored in the intermediate register area for use in the third clock period of execution. This exponent is formed from the selected 60 bit word by removing the packed format bias and complementing the exponent field if the selected word was negative. The exponent field for the other operand is discarded at this point in the process.

The coefficient for the reference operand is stored in the intermediate register area for use during the third clock period. This quantity retains the sign of the original word. A round bit is added to the reference coefficient if the round flag is set. This bit is equal to the complement of the reference sign bit and appears just below the least significant bit of the coefficient.

The coefficient for the shifted operand is partially aligned for addition with the reference operand coefficient before entering the intermediate register area. The lowest order three bits of the alignment shift count are interpreted by a static network which positions the 48 bit coefficient in a 56 bit intermediate register. A round bit may be added to this coefficient value if the round flag is set. This round bit is added only if the two coefficients in the original operands were both normalized, or if the two coefficients have different signs. If the round bit is added, it is set equal to the complement of the shifted coefficient sign bit, and is positioned just below the lowest order bit of the coefficient in the 56 bit register. The remaining positions in the 56 bit register are filled with shifted coefficient sign bits.

The alignment of the shifted coefficient for addition with the reference coefficient is completed during the third clock period of instruction execution. A static network receives the 56 bits of shifted coefficient from the intermediate register area and shifts this quantity by an additional 8, 16, 32, 64, or combination of these values of bit positions. The resulting quantity is sign extended to fill a 99 bit field.

If the shift count required for alignment of the coefficients is 128 bit positions or greater, the output of the shifting network is blocked and all zeros are delivered to the following adder. This is equivalent to shifting off the end of the field of the adder except for one special situation which can cause an anomaly in the results from the floating add unit. If the shifted coefficient is negative, the higher order bits in the 99 bit field are replaced with sign bits as the coefficient is shifted right by the amount of the shift count. The value of the coefficient and the amount of shift may be such as to leave the 99 bit field filled with sign bits for a negative zero value. If the shift count is 128 or more, the result is blocked and the 99 bit field is filled with zeros for a positive zero value. The difference between the positive zero value and a negative zero value is unimportant unless the reference coefficient is also a negative zero value. If both coefficients are negative zero the result is negative zero. If either is positive zero the result is positive zero. The result of adding two floating point numbers, both negative, and the larger with a zero coefficient, may be either a positive zero coefficient value or a negative zero coefficient value depending on the amount of the exponent difference.

The 49 bits of reference coefficient data are sign extended during the third clock period of instruction execution to fill a 99 bit field. The reference coefficient is positioned in the upper half of the field,

and the lower half is filled with reference coefficient sign bits. This data is then merged with the corresponding field of data from the shifted coefficient in a 99 bit ones complement adder. A partial sum is formed prior to the output register for the result coefficient. The 99 partial sum bits and the 99 possible carry bits are held in the output register area for use during the fourth clock period of instruction execution.

The reference exponent is transmitted from the intermediate register area to the output register area during the third clock period of instruction execution. This transmission is through a static network which modifies the exponent value if the double precision flag is set. The exponent value is reduced by 48 decimal in a 12 bit ones complement mode if the double precision flag is set. If the double precision flag is not set the exponent value is copied unaltered from the intermediate register to the output register area.

During the fourth clock period of instruction execution the results of a floating add unit computation are transmitted to the destination X register. There are four data paths from the coefficient portion of the floating add unit to the X registers. One of these four paths is selected on the basis of the instruction mode and the character of the coefficient sum. The upper portion of the 99 bit double precision sum is transmitted to the X register data input path on instructions 30, 31, 34, and 35. The lower portion of the 99 bit sum is transmitted to the X register data input path on instructions 32 and 33. A further selection is necessary because of the possibility of coefficient overflow in the double precision addition. If the sum overflows the highest order bit position occupied by the reference coefficient, an alternate output to the destination X register is selected in which the result coefficient bits are taken from the 99 bit field one bit position higher than if no overflow occurs. The combination of double precision mode, and possible coefficient overflow, require the four independent data paths.

There are four possible exponent values to be selected corresponding to the four coefficient values. The correction for double precision mode is made on the exponent during the third clock period of instruction execution. The remaining possible exponent values depend on the sign of the result coefficient and the possible overflow of the coefficient field. These conditions are resolved by a static network between the output register for the exponent and the destination X register data input path. There are two transmission paths for this information. Path selection depends on the existence of the coefficient overflow. The sign correction and packed floating point bias correction are handled in the static network.

Go floating add flag

The go floating add flag is set in the floating add unit at the end of a clock period in which a floating add instruction issues from the CIW register. This flag is copied to other ranks of registers in the floating add unit as the data progresses through the unit. The presence of the copied version of this flag during the fourth clock period of instruction execution causes the data in the floating add unit output register to be gated to the destination X register. If this flag is not set the associated data is discarded. The condition for setting the go floating add flag is as follows:

Condition: "go issue" & "registers free" & g = 3 & h = 0,1,2,3,4,5

Floating add round flag

The floating add round flag controls the rounding of the coefficients for the data in the floating add unit. This flag is set along with the data in the floating add unit input register when an instruction requiring rounding issues from the CIW register. This flag is copied to another register rank as the data moves through the unit. The condition for setting the floating add round flag is as follows:

Condition: h = 4,5,6,7

Floating add double precision flag

The floating add double precision flag controls the selection of coefficient outputs from the floating add unit and the exponent correction associated with the double precision mode. This flag is set along with the data in the floating add unit input register area when a double precision instruction issues from the CIW register. This flag is copied to other register ranks as the data moves through the unit. The condition for setting this flag is as follows:

Condition: h = 2,3,6,7

Special cases

A number of special cases of operation are sensed by the floating add unit. One category of these special cases involves overflow and indefinite operand values. These situations are sensed during the second clock period of instruction execution. The normal output from the floating add unit to the destination X register is blocked for these cases. The proper special format for the floating add result is determined in the X register input control portion of the CPU rather than in the floating add unit. The static network outputs indicating overflow or indefinite result are transmitted from the floating add unit to the X register input control during the third clock period of instruction execution. The formation of the proper word delivered to the destination X register is then performed in the X register input control unit.

A second type of special case occurs if the double precision exponent correction in the third clock period of instruction execution causes underflow of the floating point exponent range. This case is treated much like the special case operand tests in that a static network output goes to the X register input control unit indicating underflow. The output of the floating add unit is blocked in the fourth clock period of instruction execution, and the X register input control unit generates the resulting special format word.

Execution timing

The timing charts for instructions executed in the floating add unit are listed individually for each instruction in part three of this manual. The detail execution of the various portions of the floating add unit are the same for each of these instructions. The timing of these portions is indicated in the special timing chart below.

CP00 Floating add instruction issues from the CIW register.
Transmit (Xj) to the floating add unit input register.
Transmit (Xk) to the floating add unit input register.
Set the go floating add flag.

CP01 Form the difference of the operand exponents.
Select the reference operand and the shifted operand.
Perform initial coefficient alignment.

CP02 Complete the coefficient alignment.
Form partial coefficient sum in double precision mode.
Perform the double precision exponent correction if required.

CP03 Complete the double precision coefficient sum.
Transmit result to destination X register.

Floating multiply unit

The floating multiply unit executes the three CPU instructions, 40, 41, and 42. The organization of this unit is illustrated in figure 2-8 on the following page. This unit differs from the functional units previously described in that the data does not flow continuously through the unit with new data entering each clock period. Data may enter the floating multiply unit every clock period until a multiply instruction issues from the CIW register. Inputs to this unit are blocked in the clock period following issue. Inputs resume again two clock periods after instruction issue. The maximum rate at which instructions may be executed in the floating multiply unit is, therefore, one instruction every other clock period.

All instructions performed in the floating multiply unit require five clock periods for execution. Data moves from the operating registers to the multiply unit input registers in the same clock period in which a floating multiply instruction issues fro ! the CIW register. The input registers in this unit are of the clear/enter type but are not cleared automatically at the end of every clock period. The data in the input registers is cleared and new data entered whenever the multiply busy flag is cleared. When the multiply busy flag is set, the data in the input registers is held over into the following clock period. This data then resides in the input register for a total of two clock periods. These are the second and third clock periods of instruction execution.

The two operands, (Xj) and (Xk), are individually complemented on the way to the multiply unit input registers if their sign is negative. The input registers then hold only positive operand values. In addition, the input registers hold control information to complete the execution of the instruction. The signs of the operands are merged in a static network which forms the logical difference of the two sign bits. The result is stored in the input register area as the sign of the resulting product. Two bits of the h designator are stored in the input register area to determine mode of the unit as the data progresses through the unit. The multiply busy flag serves the purpose of a go multiply flag as well as blocking further entry to the unit in the clock period following issue. This multiply busy flag is copied from register rank to register rank as the data moves through the unit. This flag serves as the basic timing control which gates data into the clear/enter registers of the unit at the proper time.

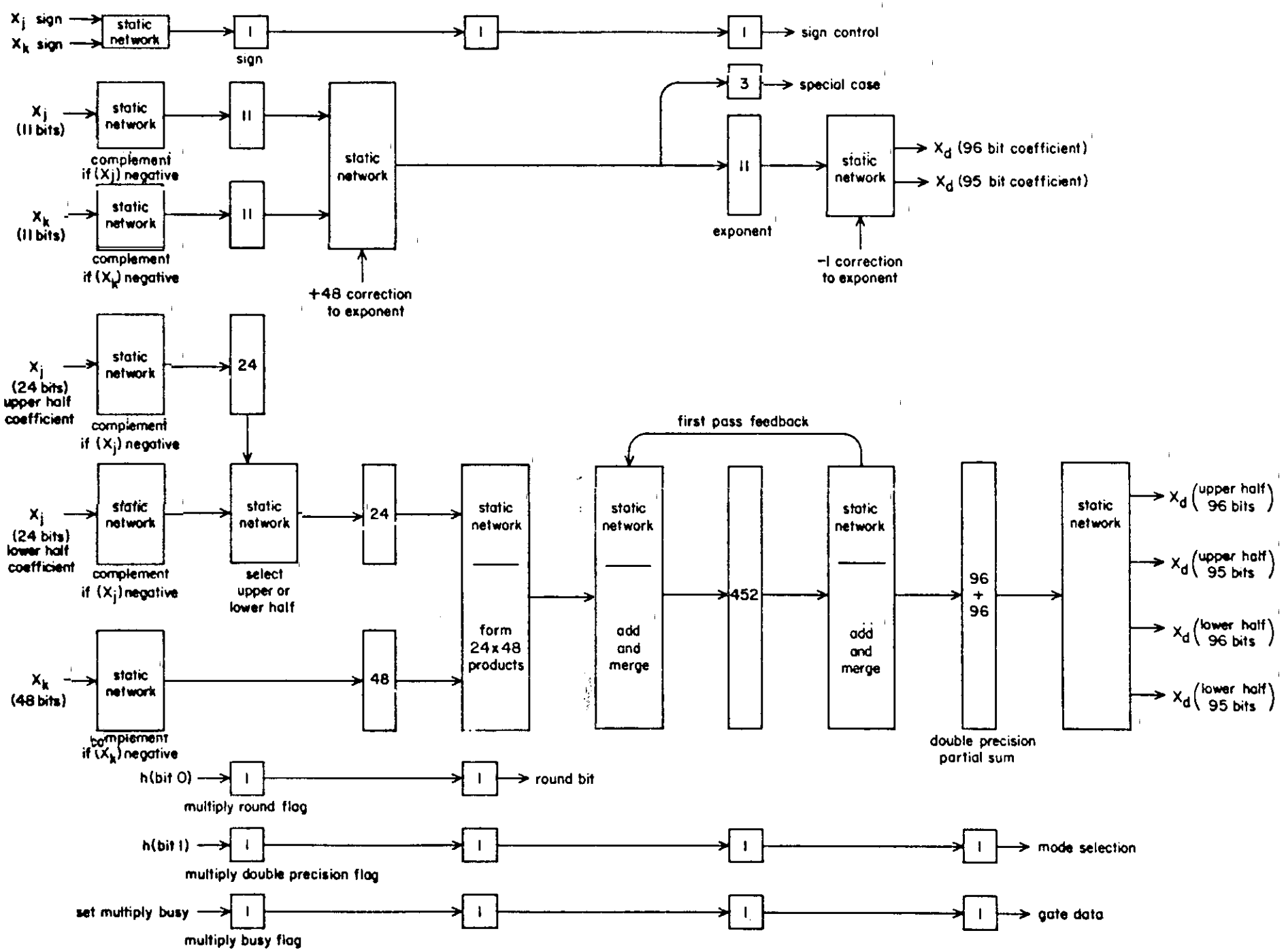


Fig. 2-8 Floating Multiply Unit

The floating multiply unit forms a 96 bit double precision product from the two operand coefficients. This product is formed in two steps. Each step utilizes 24 bits of the Xj coefficient and all 48 bits of the Xk coefficient. The process may be indicated mathematically as follows:

Let (Xj) coefficient magnitude = $\sum_{i=0}^{47} a_i \, 2^i$ where $a_i = (0,1)$

Let (Xk) coefficient magnitude = $\sum_{j=0}^{47} b_j \, 2^j$ where $b_i = (0,1)$

Then the desired product = $\sum_{i=0}^{47} a_i \, 2^i$ times $\sum_{j=0}^{47} b_j \, 2^j$

$$\sum_{i=0}^{23} \sum_{j=0}^{47} a_i \, b_j \, 2^{i+j} + \sum_{i=24}^{47} \sum_{j=0}^{47} a_i \, b_j \, 2^{i+j}$$

Each step in the two step process involves forming 1152 binary products consisting of one bit from the Xj coefficient and one bit from the Xk coefficient. These 1152 bits of data must then be added in the proper groupings to form a combined sum.

The lower half of the Xj coefficient and the complete Xk coefficient are merged during the second clock period of instruction execution. During this clock period the feedback path indicated in figure 2-8 is blocked. One static network forms the 1152 bit products, and a second static network partially sums these products. The resulting partially merged data is stored in a 452 bit register for use during the third clock period of instruction execution.

During the second clock period of instruction execution a static network selects the upper half of the Xj coefficient and delivers this data to the input register currently holding the lower half of the Xj coefficient. At the end of the second clock period this 24 bit input register is cleared and entered with the upper half data. This is the only register data in the input register area which is altered at this time.

During the third clock period of instruction execution the upper half of the Xj coefficient is merged with the complete Xk coefficient. During this clock period the feedback path indicated in figure 2-8 permits the data formed in the previous clock period to merge with the new data. This feedback data is merged with a 24 bit offset so that the bit positions of the two data fields are properly aligned. A round bit is added to bit position 46 of the merged data if the multiply round flag is set. The combined partial sum of product bits is entered in the 452 bit register for use in the fourth clock period of instruction execution.

During the fourth clock period of instruction execution the data in the 452 bit register is further summed and merged until there remains but two bits of data in each bit position of the 96 bit double precision sum. This data is delivered to two 96 bit output registers for use during the fifth clock period of execution. In this last clock period the addition process is completed, and the 96 bit product is available for transmission to the destination X register.

The two 11 bit operand exponents are processed in a separate portion of the floating multiply unit. The exponents are complemented during the first clock period of execution if the sign of the associated coefficient is negative. The packed floating point format bias is also removed from the exponents in this clock period by complementing the highest order exponent bit. The two resulting 11 bit ones complement exponents are stored in the input register area for use during the second and third clock periods of instruction execution.

During the second and third clock periods of instruction execution the exponents are held in the input registers. A static network adds the two exponents in a 13 bit ones complement mode. In addition, this static network adds a third quantity which is dependent on the instruction mode. If the multiply double precision flag is set this third quantity is zero. If the multiply double precision flag is cleared the third quantity has a value of +48 decimal. This exponent correction is necessary to compensate for the truncation of the integer coefficient product in a single precision mode. At the end of the third clock period the results from this static network are entered in an 11 bit output register for use during the fourth and fifth clock periods. The upper two bits in the 13 bit ones complement sum are interpreted for overflow or underflow of the floating point exponent range. This information is stored in a separate control register area to handle special case results.

During the fifth clock period of instruction execution the results of the coefficient calculation and the exponent calculation are transmitted to the destination X register. There are four data paths from the coefficient portion of the floating multiply unit to the X register. One of these four paths is selected on the basis of the instruction mode and the magnitude of the result coefficient. If the sign of the result is negative the entire 60 bit word is complemented at the input to the X register. This function is performed by the X register input control unit rather than the multiply unit. The sign control bit is transmitted from the multiply unit to the X register input control unit during the fourth clock period of instruction execution.

Selection of the multiply coefficient output path is a function of the mode of the instruction. If the double precision flag is cleared the upper half of the double precision result is transmitted. If the double precision flag is set the lower half of the double precision result is transmitted. A further selection of output path is based on the number of significant bits in the double precision result. If the result has 96 bits of significance the instruction mode selects between the upper 48 bits or the lower 48 bits. If the result has 95 bits of significance, and the operand coefficients were both normalized, the instruction mode selects between the most significant 48 bits or the lower 47 bits with a padded zero bit. This alternate set of data paths insures that normalized operands result in a normalized product. If the operands were not both normalized the result is treated as if there were 96 bits of significance.

There are two data paths from the multiply unit exponent area to the destination X register. Selection of the data path is made on the basis of the number of significant bits in the coefficient. The instruction mode correction on the result exponent is completed during the second and third clock periods of execution. A static network forms an exponent value for a 96 bit coefficient and an alternate exponent value for a 95 bit coefficient. This static network also corrects each exponent value for the packed floating point format bias. The selection of the two exponent values occurs during the fifth clock period of instruction execution and corresponds to the selection of the coefficient output path.

Multiply busy flag

The multiply busy flag is set in the floating multiply unit at the end of a clock period in which a floating multiply instruction issues from the CIW register. This flag is copied to other register ranks as the data is processed by the multiply unit. This flag blocks input to the multiply unit in the clock period immediately following instruction issue. This flag, and copies of this flag, control the data movement through the multiply unit. A copy of this flag controls the transmission of data from the multiply unit to the destination X register during the fifth clock period of instruction execution. The condition for setting this flag is as follows:

Condition: "go issue" & "registers free" & g = 4 & h = 0,1,2

Multiply double precision flag

The multiply double precision flag controls the mode of instruction execution in the floating multiply unit. This flag is set along with the data in the multiply unit input register area when a double precision multiply instruction issues from the CIW register. This flag is copied to other register ranks in the multiply unit as the data is processed. The condition for setting this flag is as follows:

Condition: h = 2,3,6,7

Multiply round flag

The multiply round flag controls the addition of a round bit to the double precision coefficient during the third clock period of instruction execution. This flag is set during the first clock period of instruction execution and is copied to another register rank for use during the third clock period of execution. The condition for setting the multiply round flag is as follows:

Condition: h = 1,3,5,7

Special cases

A number of special cases of operation are provided in the floating multiply unit. One category of special cases involves overflow, underflow, or indefinite operand values. These situations are sensed during the third clock period of instruction execution, and special case flags are set to block the normal output from the multiply unit to the destination X register during the fifth clock period of execution. These special case flags are copied to the X register input control unit during the fourth clock period of instruction execution. The formation of the proper word delivered to the destination X register is then performed in the X register input control unit.

A second type of special case occurs during the third clock period of execution if the exponent arithmetic results in an overflow, or an underflow, of the floating point exponent range. The special case flag is set in this case, and the word delivered to the destination X register is formed in the X register input control unit.

Execution timing

The timing charts for instructions executed in the floating multiply unit are listed individually in part three of this manual. The timing of the data movement through the multiply unit is the same for each of these instructions. This common timing sequence is listed below:

CP00 Floating multiply instruction issues from the CIW register.
Transmit (Xj) to the multiply unit input register.
Transmit (Xk) to the multiply unit input register.
Set the multiply busy flag.

Form the first 24 by 48 bit product.
Begin exponent arithmetic.
Hold the data in the input register area.
Clear the multiply busy flag.

CP02 Form the second 24 by 48 bit product.
Partially merge the data from the previous clock period.
Sense special case exponent values.
Complete exponent arithmetic.

Complete merge of coefficient data to two bits per position.
Form alternate exponent value.

CP04 Form 96 bit coefficient value.
Select data path for transmission to X register.
Transmit result from multiply unit to destination X register.

Floating divide unit

The floating divide unit executes the two CPU instructions, 44 and 45. The organization of this unit is illustrated in figure 2-9 on the following page. This unit involves a seventeen step iterative process to form the quotient from the two operands. Only one divide instruction may be executed in the iterative portion of the divide unit at a given time.

The divide instructions require 20 clock periods for execution. Data moves from the operating registers to the divide unit input registers each clock period in which the divide busy flag is cleared. This data is used for instruction execution only if a divide instruction issues from the CIW register and sets the divide busy flag. The divide busy flag blocks the inputs to the divide unit in the 17 clock periods following issue of a divide instruction. The data which arrived at the divide unit during the clock period of instruction issue is then used in the execution of the following divide sequence. A second divide instruction may issue from the CIW register 18 clock periods after the previous divide instruction.

The divide unit input registers are of the clear/enter type but are not cleared automatically at the end of each clock period. The data in the input registers is cleared and new data entered only at the end of those clock periods in which the divide busy flag is cleared. When the divide busy flag sets the data in the input registers is held over for the following 17 clock periods. One bit of the h designator is held in the input register area along with the operand data (Xj) and (Xk). This bit is the divide round flag which distinguishes between the two instruction modes.

The divide unit operates on positive coefficient values only. The coefficients for (Xj) and (Xk) are individually complemented in static networks if their sign is negative. The sign of the result is determined at the input to the destination X register by the X register input control unit. This sign bit is the logical difference of the two operand sign bits.

The divide busy flag initiates a chain of divide sequence control flags which sequence the remainder of the steps in the instruction execution. This sequence control provides a static condition to distinguish each of the 19 clock periods following the issue of the divide instruction. These static conditions then control the data movement within the divide unit and the data transmission to the X register input path at the end of the divide sequence.

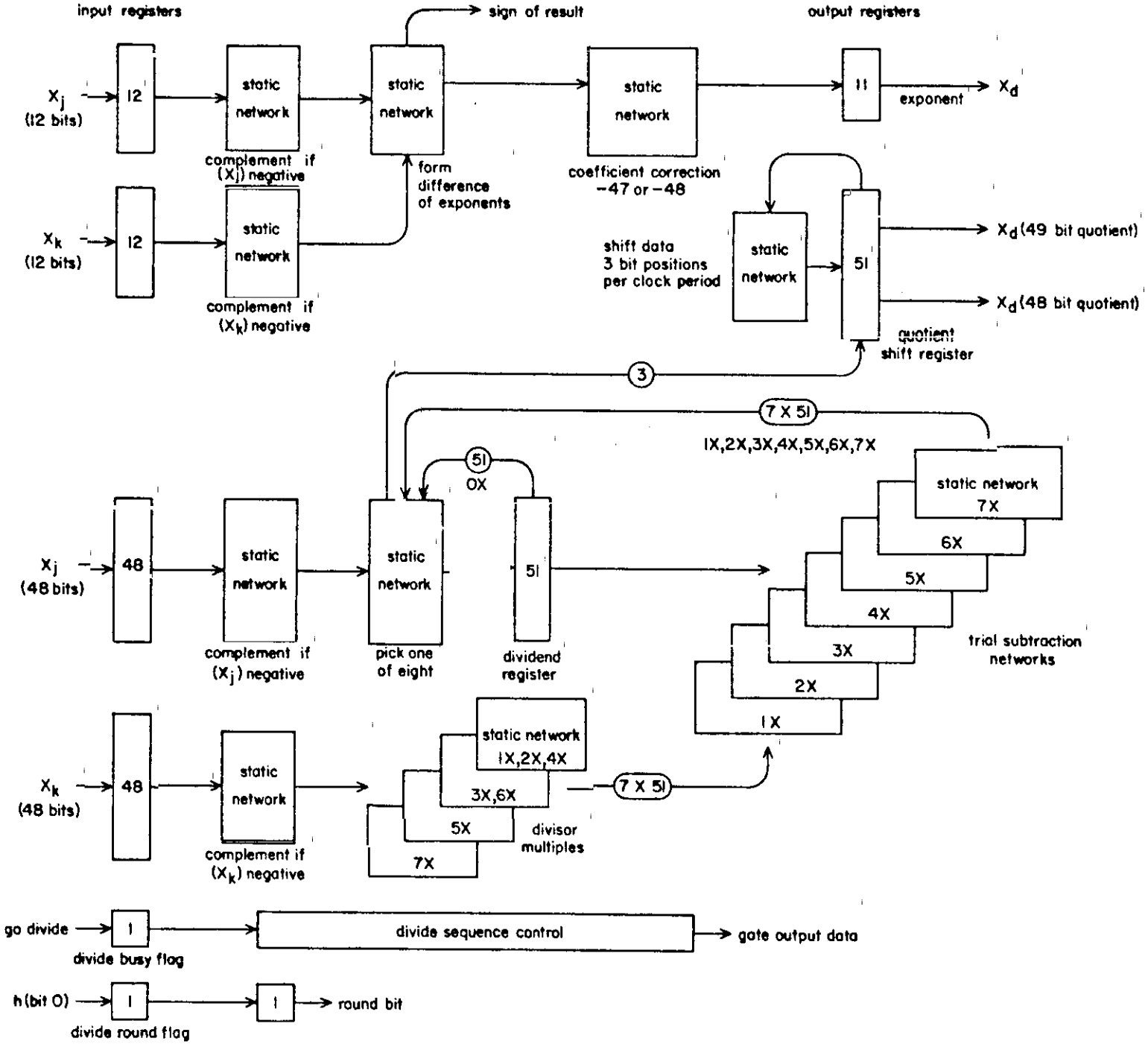


Fig. 2-9 Floating Divide Unit

The dividend for the divide sequence is the coefficient magnitude of (Xj). This quantity is entered in a 51 bit dividend register during the second clock period of instruction execution. The dividend register is a clear/enter type register which is cleared at the end of each clock period. The initial 48 bit dividend value is entered in the lower 48 bits of the dividend register. The upper three bits of this register are entered with zero bits.

The divisor for the divide sequence is the coefficient magnitude of (Xk). This quantity is delivered to four static networks which form multiples of the divisor. Each multiple of the divisor from one through seven is formed. Multiplication by one, two, and four is accomplished by shifting the bit positions of the divisor value. Multiplication by six is similar to the multiplication by three, with a one bit shift of data. These seven sets of numbers are held statically throughout the 17 step divide sequence.

The quotient from the divide sequence is determined three bits per clock period. This is accomplished by trial subtracting the seven multiples of the divisor from the value currently in the dividend register. One of these seven values, or the previous dividend register value, is then selected for entry in the dividend register at the end of each clock period. The choice between the eight possible values is made on the basis of the signs of the trial differences. The largest multiple of the divisor is picked which will cause the new dividend value to be positive. The new dividend value is entered in the dividend register with a three bit position shift. The lowest order three bits in the dividend register are entered with zero bits. The three bits of quotient data are entered in a quotient shift register at the end of each clock period. This shift register is a clear/enter type register which is cleared at the end of each clock period. The data is re-entered with a three bit position shift at the end of each clock period. The three bits of new quotient data enter the lowest order three bit positions in the quotient shift register. At the end of 17 iterative steps there are 51 accumulated bits of quotient data.

The operand exponents are processed in a separate section of the divide unit. The upper 12 bits of (Xj) and (Xk) are held in the divide unit input register area for the 17 clock periods of the iterative process. A static network is provided for each of the exponents to complement the exponent value if the sign of the operand is negative. These static networks also remove the packed floating point format bias by complementing the highest order exponent bit. The logical difference of the sign bits is delivered to the X register input control unit to specify the sign of the result.

The operand exponents are subtracted in a 13 bit ones complement mode. The Xk exponent is subtracted from the Xj exponent. A correction is then made for the integer coefficient in the resulting quotient representation. This correction is either -47 or -48 depending on the number of significant bits in the integer coefficient for the quotient. The resulting exponent value may overflow or underflow the floating point exponent range. The upper two bits of the 13 bit exponent are used to sense these conditions and set an appropriate special case flag. The lower order 11 bits are delivered to the divide unit output register at divide time 18. This corresponds to the 19th clock period of instruction execution. In the 20th, or last, clock period of instruction execution the data is transmitted from the divide unit output registers to the X register data input path.

The result coefficient is assembled three bits per clock period in the quotient shift register. This register also serves as the coefficient output register. There are two sets of data paths from this register to the X register data input path. The first three bits to enter the 51 bit quotient shift register contain the information necessary to make the exponent correction and choose the output transmission path. If this first octal digit has a value of zero, the lower order 48 bits of the 51 bit shift register will be transmitted to the destination X register. If the octal digit has a value of one, the 48 bits transmitted to the X register will originate one bit position higher in the 51 bit register. The lowest order bit of the computed quotient will be discarded in this case. If the octal digit has a value greater than one, the divide sequence result will be treated as a special case, and the indefinite condition flag will be set for interpretation by the X register input control unit.

There is only one data transmission path from the exponent portion of the divide unit output register to the X register data input path. The exponent correction corresponding to the 48, or 49, bit quotient coefficient is made prior to the data entry in the exponent output register. The exponent bias for packed floating point format is added in the transmission path from the divide unit to the destination X register. The sign of the result is determined by the X register input control unit. The entire 60 bit word is complemented at the input to the X register if the result is negative.

Divide busy flag

The divide busy flag is set in the floating divide unit at the end of a clock period in which a divide instruction issues from the CIW register. This flag remains set in the following 17 clock periods. It is cleared by the divide sequence control at divide time 17. This flag holds the data in the divide unit input registers for the 17 clock periods required for the iterative divide sequence. The conditions for setting and clearing this flag are as follows:

Set condition: "go issue" & "registers free" & g = 4 & h = 4,5

Clear condition: "divide time 17"

Divide round flag

The divide round flag is set in the floating divide unit at the end of a clock period in which a round floating divide instruction issues from the CIW register. This flag modifies the dividend in the third clock period of instruction execution. An octal digit with a value of four is entered in the lowest order bits of the dividend register if this flag is set. This occurs only for the one clock period. Zero bits are entered in these bit positions during all other clock periods in the divide sequence. This modification of the dividend has the effect of increasing the dividend value by one half of the least significant bit in the original operand. The condition for setting the divide round flag is as follows:

Condition: h = 1,3,5,7

"Divide time 13" condition

This condition originates in the divide sequence control and is used in the SCM access control. This static condition exists during the 14th clock period of execution for a divide instruction. It is used in the SCM access control to block initiation of a SCM storage reference that would conflict with the delivery of data from the divide unit to the X register data input path.

"Divide time 15" condition

This condition originates in the divide sequence control and is used in instruction issue control and in the X register access control unit. This static condition exists during the 16th clock period of execution for a divide instruction. It is used in the instruction issue control to block issue of an instruction which would conflict with the delivery of data from the divide unit to the X register data input path. It is used in the X register access control to initiate the process of register access for the destination X register.

"Divide time 17" condition

This condition originates in the divide sequence control and is used to clear the divide busy flag and enter the exponent value in the output register. This static condition exists during the 18th clock period of execution for a divide instruction.

Special cases

A number of special cases are treated in the floating divide unit. One such category involves overflow, underflow, or indefinite operand values. These situations are sensed in the static network which forms the difference of the exponent values. The combinations of special operand values which cause specific results are listed in part three of this manual for the individual instructions. The static condition specifying a special case result is delivered to the X register access control unit. The transmission of data from the divide unit output registers to the destination X register is blocked for these cases. The special case word is then formed in the X register access control.

A second category of special cases occurs if there is an underflow, or an overflow, of the floating point exponent range during the exponent calculation in the divide unit. In these cases the special case result is indicated to the X register access control unit, and the output from the divide unit is blocked in the same manner as for the special case operands.

A third special case category occurs if the initial trial subtraction in the coefficient calculation results in an octal digit with a value of 2 or more. This is a divide fault situation which occurs if the divisor is not normalized. The indefinite condition result is indicated for this case, and it is treated in the same manner as the other special cases.

Execution timing

The timing charts for the two divide instructions are listed in part three of this manual. These two charts are essentially the same, and the common portion related to the internal timing of the divide unit is listed below in somewhat greater detail.

CP00 Divide instruction issues from the CIW register.
Transmit (Xj) to the divide unit.
Transmit (Xk) to the divide unit.
Set divide busy flag.

CP01 Transmit magnitude of Xj coefficient to the dividend register.
Begin translation of divisor multiples.
Begin exponent calculation.

CP02 First trial subtraction.
Transmit picked value to the dividend register.
Transmit round bit (if any) to the dividend register.
Transmit first octal digit to quotient shift register.

CP03 Second trial subtraction.
Transmit picked value to the dividend register.
Transmit second octal digit to the quotient shift register.

CP04 Third trial subtraction.
Transmit picked value to the dividend register.
Transmit third octal digit to the quotient shift register.

Fourth trial subtraction.
Transmit picked value to the dividend register.
Transmit fourth octal digit to the quotient shift register.

CP06 Fifth trial subtraction.
Transmit picked value to the dividend register.
Transmit fifth octal digit to the quotient shift register.

Sixth trial subtraction.
Transmit picked value to the dividend register.
Transmit sixth octal digit to the quotient shift register.

CP08 Seventh trial subtraction.
Transmit picked value to the dividend register.
Transmit seventh octal digit to the quotient shift register.

Eighth trial subtraction.
Transmit picked value to the dividend register.
Transmit eighth octal digit to the quotient shift register.

CP10 Ninth trial subtraction.
Transmit picked value to the dividend register.
Transmit ninth octal digit to the quotient shift register.

Tenth trial subtraction.
Transmit picked value to the dividend register.
Transmit tenth octal digit to the quotient shift register

Eleventh trial subtraction.
Transmit picked value to the dividend register.
Transmit eleventh octal digit to the quotient shift register.

Twelfth trial subtraction.
Transmit picked value to the dividend register.
Transmit twelfth octal digit to the quotient shift register.
Transmit block SAS issue signal to SCM control.

CP14 Thirteenth trial subtraction.
Transmit picked value to the dividend register.
Transmit thirteenth octal digit to the quotient shift register.

Fourteenth trial subtraction.
Transmit picked value to the dividend register.
Transmit fourteenth octal digit to the quotient shift register.
Block issue of multiply instructions.
Initiate X register access control.

CP16 Fifteenth trial subtraction.
Transmit picked value to the dividend register.
Transmit fifteenth octal digit to the quotient shift register.

CP17 Sixteenth trial subtraction.
Transmit picked value to the dividend register.
Transmit sixteenth octal digit to the quotient shift register.
Clear divide busy flag.
Transmit exponent value to the output register.

CP18 Seventeenth trial subtraction.
Transmit picked value to the dividend register.
Transmit seventeenth octal digit to the quotient shift register.

Transmit result from divide unit to destination X register.

Population count unit

The population count unit executes CPU instruction 47. This instruction counts the number of bits in the 60 bit operand which have a value of one. There is only one mode of operation for this unit. The organization of the data paths is illustrated in figure 2-10 on the following page.

The population count instruction requires two clock periods for execution. Data moves from the operating register to the population count unit in the same clock period in which the instruction issues from the CIW register. Data moves from the population count unit back to the operating registers during the following clock period. A new instruction may be issued for execution in the population count unit each clock period.

The contents of the Xk register are transmitted to the population count unit each clock period. This data enters a static network which partially sums the one bits and enters the partially reduced data in a 27 bit input register. This data is used during the following clock period only if the go pop count flag is set. If the go pop count flag is set, the count is completed in a static network following the input register. The resulting six bits of count data plus 54 bits of zero data are then transmitted to the destination X register. The count is transmitted in the lower order six bit positions in the word.

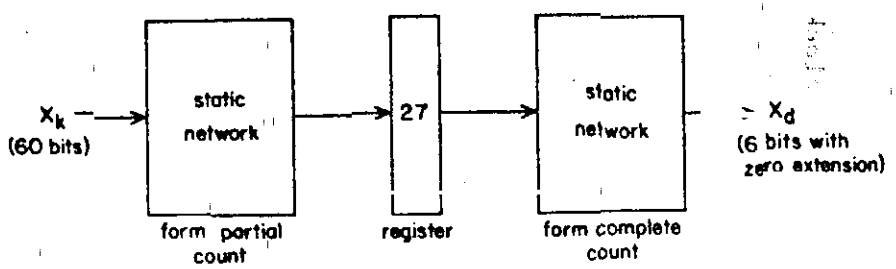
The 27 bit input register for the population count unit is a clear/enter type register which is cleared at the end of each clock period.

Go pop count flag

The go pop count flag is set in the population count unit at the end of a clock period in which a 47 instruction issues from the CIW register. This flag then controls the transmission of data from the population count unit to the destination X register. If the go pop count flag is not set during a given clock period, the data in the population count unit is discarded. The condition for setting the go pop count flag is as follows:

Condition: "go issue" & "register free" & g = 4 & h = 7

The timing chart for the 47 instruction is listed in part three of this manual.



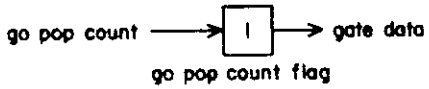


Fig. 2-10 Population Count Unit

Increment unit

The increment unit executes CPU instructions 50 through 77. These instructions involve address length arithmetic. This unit is somewhat special in that those instructions with an A register destination cause a SCM storage reference in addition to the arithmetic computation. The address delivered to the SAS for the SCM storage reference is computed in a separate portion of the increment unit. The organization of this unit is illustrated in figure 2-11 on the following page.

The increment instructions require two clock periods for execution. Data moves from the operating registers to the increment unit in the same clock period in which the instruction issues from the CIW register. Data moves from the increment unit to the destination operating register during the following clock period. A new instruction may be issued for execution in the increment unit each clock period.

The increment unit contains a single level of registers for data and control information. These registers are all of the clear/enter type, and are cleared at the end of every clock period. A portion of the arithmetic operation is performed in static networks prior to the entry of data in the increment registers.

Data arrives at the increment unit from five different input paths. One group of data paths consists of inputs from the Xj, Bj, and Aj registers. A static network selects one of these three data paths for an increment operand. This selection is determined by the value of the h designator in the CIW register. A second static network selects the second operand. This group consists of the K field in the CIW register, the Bk register data, and the complement of the Bk register data. This selection is also made on the basis of the h designator value. The two selected 18 bit operands are partially added, and the resulting sum bits and borrow bits are stored in increment unit registers for use during the second clock period of instruction execution.

The two selected 18 bit increment operands are partially merged with the SCM reference address in a separate static network. This computation is independent of, and proceeds in parallel with, the operation described above. A data path from the RAS register to the increment unit merges with the selected operands for the increment unit. This static network forms the partial sum of the three operands simultaneously. This information is held in two 18 bit registers for use during the second clock period of instruction execution.

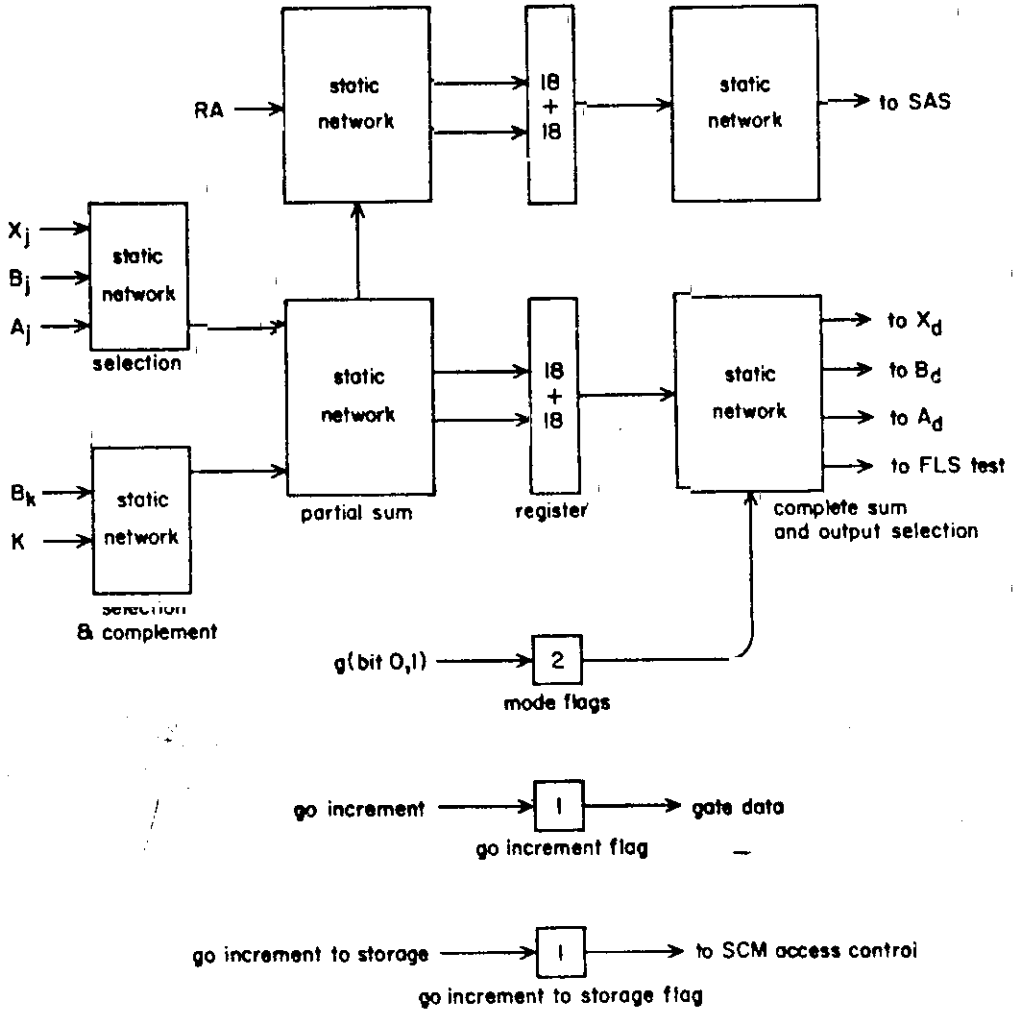


Fig. 2-11 Increment Unit

The two portions of the increment unit complete the two arithmetic calculations during the second clock period of instruction execution. One static network completes the addition of the two selected operands for the destination operating register. The output of this network is then transmitted to the required register input path under the control of the go increment flag and two increment unit mode flags. A second static network completes the addition of the two selected operands plus the SCM reference address. This network output data is transmitted to the SAS under control of the go increment to storage flag.

Go increment flag

The go increment flag is set in the increment unit at the end of a clock period in which an increment instruction issues from the CIW register. This flag then gates the data from a static network to the destination operating register during the following clock period. If the go increment flag is not set during a given clock period, the data in the increment unit during that clock period is discarded. The condition for setting this flag is as follows:

Condition: "go issue" & "registers free" & g = 5,6,7

Go increment to storage flag

The go increment to storage flag is set in the increment unit at the end of a clock period in which an instruction issues from the CIW register which requires a SCM storage reference. The go increment flag is set at this same time. The go increment to storage flag gates the data from the second portion of the increment unit to the SAS. This data transmission occurs in the same clock period as the transmission of data from the increment unit to the destination operating register. The condition for setting this flag is as follows:

Condition: "go issue" & "registers free" & g = 5 & i = 1,2,3,4,5,6,7

Increment unit mode flags

There are two mode flags in the increment unit which control the selection of destination operating register type when the go increment flag is set. These two mode flags are copies of the lowest two bits of the g designator in the instruction. If g = 5 in the executed instruction, the destination operating register is an A register. If g = 6 in the executed instruction, the destination register is a B register. If g = 7 in the executed instruction, the destination register is an X register.

There is a fourth data transmission path from the first portion of the increment unit to the error detection unit. The data transmitted to the error detection unit is the same 18 bit quantity that is transmitted to the destination operating register. This data is used to test against the SCM field length only if the increment unit causes a SCM storage reference. The data transmission occurs during every clock period, and the selection is performed in the error detection unit.

Instruction timing

Timing charts for the instructions executed in the increment unit are listed for each case in part three of this manual. The timing is complex in the case of instructions 50 through 57 where a SCM storage reference is involved. The result of the increment calculation is always delivered to the destination operating register at the end of the second clock period of instruction execution. If a SCM reference is also involved, the timing of the transmission of data from the SCM to the destination X register is under the control of the SCM portion of the CPU. The timing of this data delivery will depend on storage conflicts in SCM and will not be a constant delay.

Branch instructions

Branch instructions are executed in a portion of the CPU which is not well defined as a functional unit. This portion of the CPU contains the program address register (P), and those control flags required to sequence the entry of a new address into the P register. This portion of the CPU also contains the register and control flags required to execute the return jump instruction. These registers and flags are illustrated in figure 2-12 on the following page.

P register

The P register is logically an 18 bit register which contains the current program execution address. This register enters into a large number of functions in the CPU, and a large number of interconnecting transmission paths are required. As a result, the P register is physically reproduced a number of times in hardware to allow the proper fanout of information. Each hardware copy of the P register is entered with new data at the same clock period and under the same conditions. The P register is referenced in this manual as a single register even though multiple hardware copies exist.

The P register is a clear/enter type register with gated clock pulse control. There are four possible sources of data for entry into the P register. A static network selects one of the four data paths as illustrated in figure 2-12. The current contents of the P register are cleared, and new data is entered at the end of a clock period in which the following condition exists.

Condition: APF & no OSF
or: GJF & no "fall through"
or: SXF & NBF
or: "XJ enter P"

The selection of data for entry into the P register is made on the basis of three control conditions. Each condition specifies one of the data paths. The absence of any of the three specific conditions selects the fourth data path. This last path is the one which increases the contents of the P register by one count. The three specific control conditions are as follows:

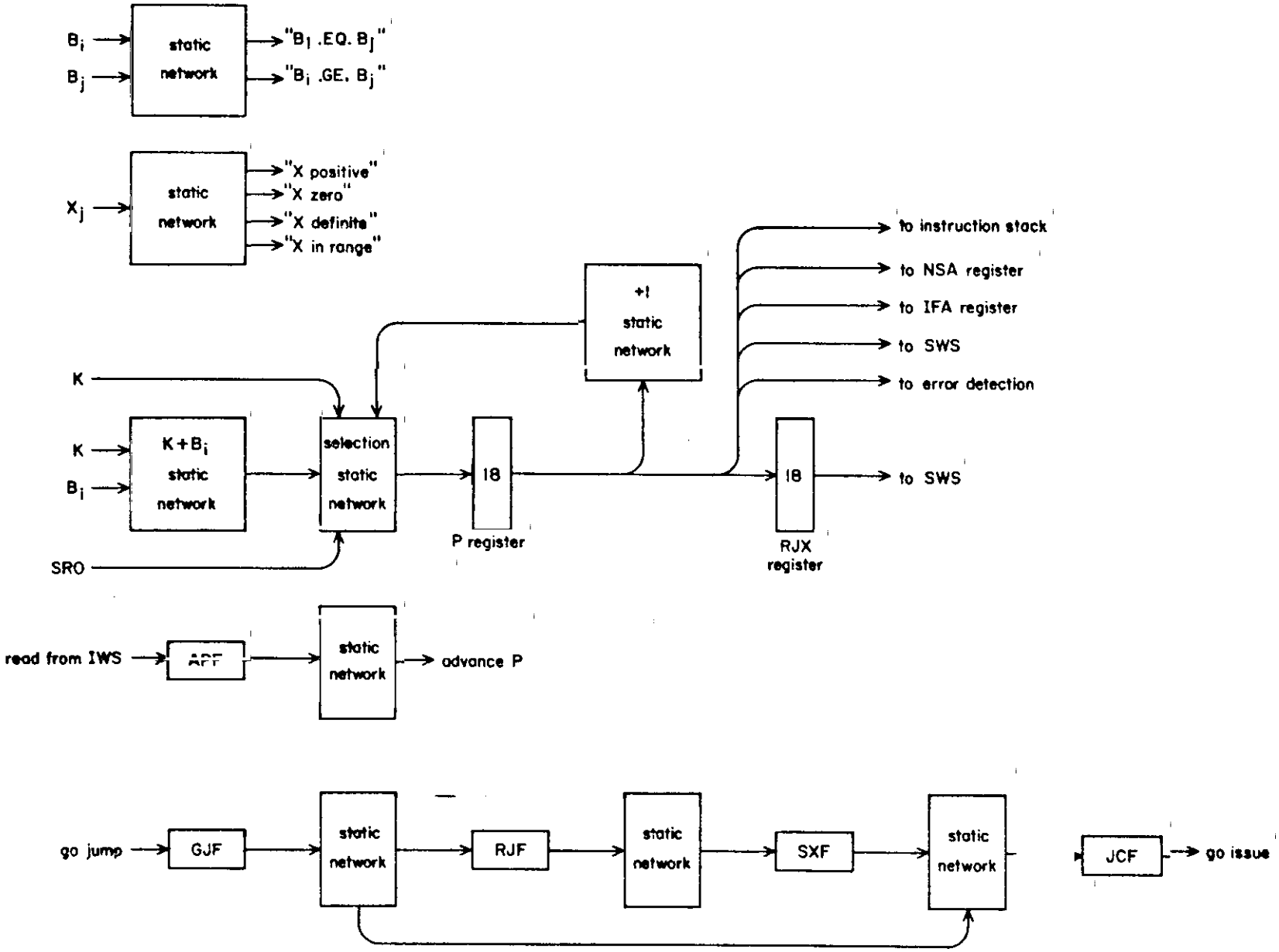


Fig. 2-12 Instruction Branching

Select the information in the K field of the CIW register for entry into the P register on the following condition:

Condition: GJF & h = 0,1,3,4,5,6,7

Select the output of a static network which forms the 18 bit ones complement sum of K + (Bi) on the following condition:

Condition: GJF & h = 2

Select bits 36 through 53 of the word currently in the SRO register for entry into the P register on the following condition:

Condition: "XJ enter P"

RJX register

The RJX register holds the return jump exit address during execution of a return jump instruction. This is an 18 bit clear/enter type register with gated clock pulse control. The current contents of the P register are entered into the RJX register at the end of a clock period in which the following condition exists:

Condition: GJF & g = 0 & h = 1 & i = 0

The information in the RJX register is transmitted to the SWS at the proper time to store the special word in SCM for the called subroutine exit. The address for the SCM storage reference is processed through the IFA register. Control for the storage reference is via the SXF. A 60 bit word is formed from the 18 bits in the RJX register. The lowest order 30 bits in this word are zeros. The next 18 bits are transmitted from the RJX register. The uppermost 12 bits are a constant instruction code for the subroutine exit. This portion of the word has an octal value of 0400.

The RJX register is not cleared after the data is transmitted to the SWS. The last return jump exit address resides in this register until the execution of the next return jump instruction.

"Fall through" condition

This control condition occurs whenever a branch instruction is executed in which the jump to a new program sequence is not taken. When this condition exists the program continues with the current program sequence. This condition is defined as follows:

```
Not:  g = 0 & h = 1 & i = 0
nor:  g = 0 & h = 2
nor:  g = 0 & h = 3 & i = 0 & "X zero"
nor:  g = 0 & h = 3 & i = 1 & no "X zero"
nor:  g = 0 & h = 3 & i = 2 & "X positive"
nor:  g = 0 & h = 3 & i = 3 & no "X positive"
nor:  g = 0 & h = 3 & i = 4 & "X in range"
nor:  g = 0 & h = 3 & i = 5 & no "X in range"
nor:  g = 0 & h = 3 & i = 6 & "X definite"
nor:  g = 0 & h = 3 & i = 7 & no "X definite"
nor:  g = 0 & h = 4 & "Bi .EQ. Bj"
nor:  g = 0 & h = 5 & no "Bi .EQ. Bj"
nor:  g = 0 & h = 6 & "Bi .GE. Bj"
nor:  g = 0 & h = 7 & no "Bi .GE. Bj"
```

"X zero" condition

This control condition is formed in a static network every clock period for use when a branch instruction is executed. This condition exists whenever all 60 bits in the Xj register are zero.

"X positive" condition

This control condition is formed in a static network every clock period for use when a branch instruction is executed. This condition exists whenever the highest order bit in the Xj register is zero.

"X definite" condition

This control condition is formed in a static network every clock period for use when a branch instruction is executed. This condition is determined by the value of the uppermost 12 bits in the Xj register. This condition exists when these 12 bits have the following octal digit values:

```
Not:  1777
nor:  6000
```

"X in range" condition

This control condition is formed in a static network every clock period for use when a branch instruction is executed. This condition is determined by the value of the uppermost 12 bits in the Xj register. This condition exists when these 12 bits have the following octal digit values.

Not: 1777
nor: 6000
nor: 3777
nor: 4000

"Bi .EQ. Bj" condition

This control condition is formed in a static network every clock period for use when a branch instruction is executed. This condition is determined by comparing the contents of the Bi and the Bj registers. This condition exists when (Bi) is identical with (Bj) on a bit by bit basis. This condition does not exist when one register contains all one bits and the other register contains all zero bits.

"Bi .GE. Bj" condition

This control condition is formed in a static network every clock period for use when a branch instruction is executed. This condition is determined by comparing the contents of the Bi and the Bj registers. This condition exists when (Bi), considered as a signed integer, is greater than, or equal to, (Bj), also considered as a signed integer. In this case a quantity consisting of all zero bits is greater than a quantity consisting of all one bits.

Advance P flag (APF)

The APF is a clear/enter type bit register which is cleared at the end of every clock period. This control flag is set at the end of a clock period in which the following condition exists:

Condition: "go issue" & "registers free" & "read stack"

This flag is used to advance the contents of the P register by one count whenever a new word is read from the IWS to the CIW register. The flag is set in the same clock period in which the data is transmitted to the CIW register. The content of the P register is advanced in the following clock period.

Go jump flag (GJF)

The GJF is a clear/enter type bit register which is cleared at the end of every clock period. This control flag is set at the end of a clock period in which the following condition exists:

Condition: "registers free" & g = 0 & h = 1 & i = 0 & no SXF & no RJF & no JCF & no GJF
or: "registers free" & g = 0 & h = 2,3,4,5,6,7 & no SXF & no RJF & no JCF & no GJF

This flag is set whenever a branch instruction is ready for execution. If the jump to a new program sequence occurs, the RJF or the JCF is set in the following clock period. If the jump is not taken, the branch instruction in the upper parcel of the CIW register is issued as a pass.

Return jump flag (RJF)

The RJF is a bit register with a separate set and separate clear input. This flag is set at the end of a clock period in which the CIW register contains a return jump instruction and the GJF is set. This flag remains set until any words requested from SCM for the IWS have arrived at the IWS. The SXF is then set and the RJF cleared. The conditions for setting and clearing this bit register are as follows:

Set condition: GJF & g = 0 & h = 1 & i = 0

Clear condition: RJF & no M1F

Store exit flag (SXF)

The SXF is a bit register with a separate set and separate clear input. This flag is set at the same time that the RJF is cleared. This flag then continues the sequence for the return jump instruction. This flag remains set until the SCM access control has accepted the address for writing the RJX register data into SCM. This flag then clears and the JCF is set. The conditions for setting and clearing this flag are as follows:

Set condition: RJF & no M1F

Clear condition: SXF & NBF
or: GJF & no "fall through" & h = 0,2,3,4,5,6,7
or: GJF & no "fall through" & i = 1,2,3,4,5,6,7

Jump completed flag (JCF)

The JCF is a clear/enter type bit register which is cleared at the end of every clock period. This control flag is set at the end of a clock period in which the following condition exists:

Condition: "go issue" & "XSK = 15"
or: SXF & NBF
or: GJF & no "fall through" & h = 0,2,3,4,5,6,7
or: GJF & no "fall through" & i = 1,2,3,4,5,6,7

This flag sequences the last step in either a normal branch instruction or a return jump instruction. The presence of this flag allows the branch instruction to issue from the CIW register and the program sequence to continue.

Normal branch execution

A normal branch instruction, as distinguished from a return jump instruction, has three cases to consider in execution timing. One case occurs when the jump criterion is not met in a conditional branch instruction. This is the "fall through" case. A second case occurs when the jump is taken and the destination address is currently within the IAS. A third case occurs when the jump is taken out of the range of the instruction stack. Each of these cases has a separate timing sequence.

All normal branch instructions begin execution with the setting of the GJF. This occurs when the conditions specified by the instruction have been resolved. These conditions involve register data which may be in process in functional units as a result of previously issued instructions. The conditions for setting the GJF are therefore similar to the conditions for issue of a computation instruction. The GJF allows the branch instruction to issue from the CIW register only in the "fall through" case. The other two cases involve further sequence control flags.

A branch instruction in which the jump criterion is met requires the entry of a new program sequence address in the P register. The P register is cleared and entered with a new address at the end of the clock period in which the GJF is set. The JCF is set at this same time to indicate that the jump has been completed. The branch instruction then issues from the CIW register in the following clock period. If the new program address is in the IAS, the next instruction word may read directly into the CIW register in this clock period. If the new program address is not in the IAS, a SCM reference must be initiated to read the new program instruction word. This process is a function of the instruction stack control.

Return jump execution

Execution of a return jump instruction begins with the setting of the GJF in a manner similar to that required for a normal branch instruction. The GJF causes two register actions in the one clock period in which it is set. The content of the P register is cleared and the new program sequence address is entered. This is the address of the exit word for the called subroutine. During this same clock period the previous address in the P register is copied into the RJX register. This is the address of the instruction word currently in the CIW register, plus one. The RJF is set at the end of this clock period to continue the sequence.

The RJF is a delay mechanism to allow any SCM instruction word references to be executed to completion. The RJF statically clears all ranks of the IAS for as long as it is set. When all instruction words previously requested have arrived at the IWS, the RJF clears and the SXF is set to continue the sequence. The contents of the P register are transmitted to the IFA register during the last clock period in which the RJF is set.

The SXF causes the SCM reference to store the special word required for the called subroutine exit. The data for this word is in the RJX register. The address for the SCM reference is in the P register. This address is also in the IFA register. The SXF remains set until the SCM reference has been initiated by the acceptance of (IFA) in the SAS. The SXF then clears and the JCF is set to complete the sequence. The content of the P register is advanced one count during the last clock period in which the SXF is set.

The JCF allows the return jump instruction to issue from the CIW register. This flag is set for only one clock period. During this clock period the P register contains the entrance address for the called subroutine. The IFA register contains this same address. No coincidence is possible between (P) and the IAS because of the previous clearing of the IAS. The instruction stack control will therefore initiate the SCM references to read up the beginning of the called subroutine.

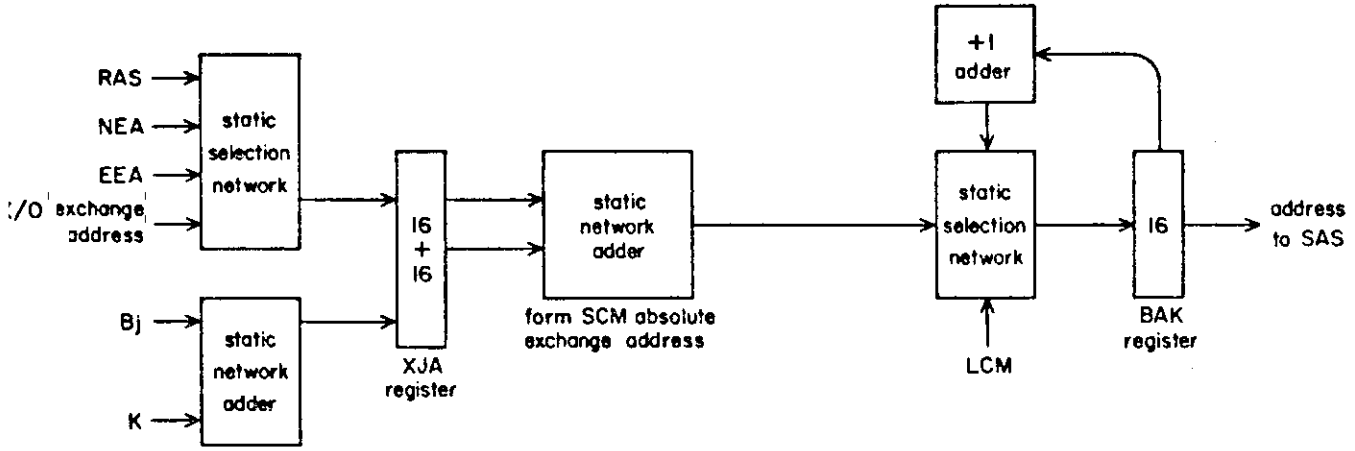
Exchange sequence

The exchange sequence involves the interchange of data between the operating registers in the CPU computation section and a package of 16 consecutive storage locations in SCM. Information from the operating registers is written into the same 16 storage locations which provide the new operating register information. The exchange of data in this sequence terminates the execution of one CPU program and begins the execution of a new CPU program.

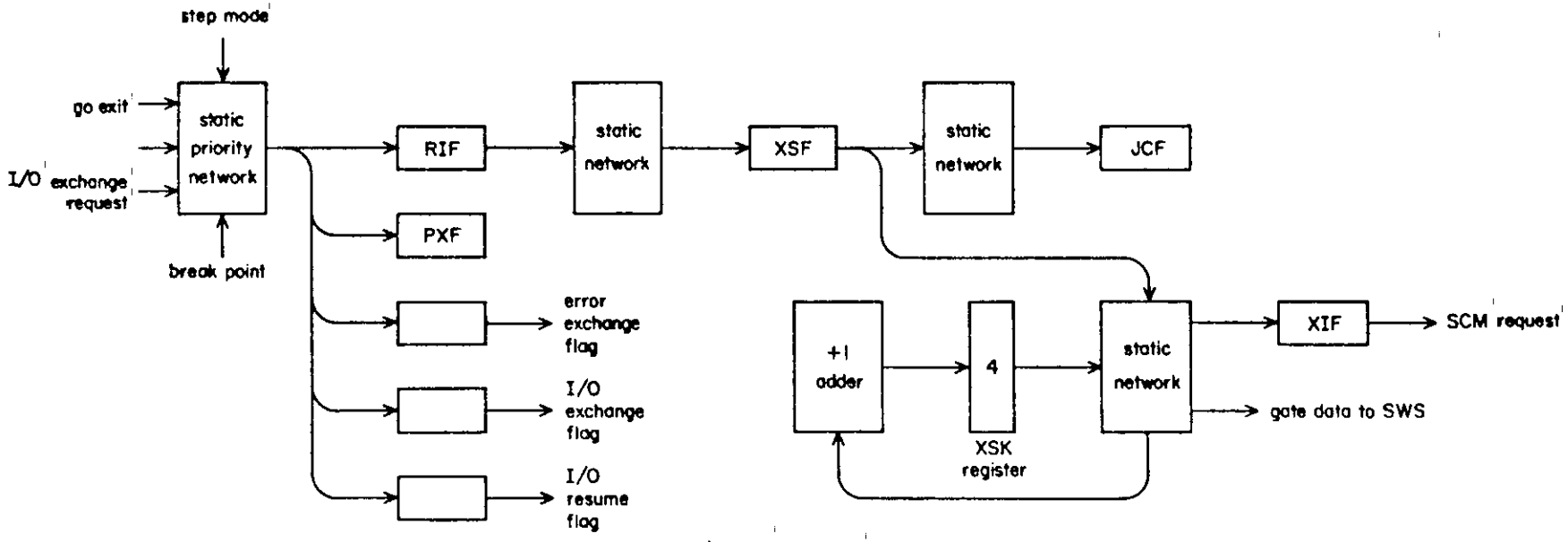
The package of data read from SCM into the CPU operating registers in the exchange sequence is called the "exchange package" for the associated CPU program. This exchange package resides in the CPU computation section throughout a period of time called the "execution interval" for the exchange package. The execution interval is terminated by another exchange sequence which returns the exchange package to SCM and initiates the execution interval for another exchange package. An exchange package for a CPU object program normally resides in a specific SCM location. The execution interval for this exchange package is initiated by the system monitor program. The monitor program exchange package then resides in the object program exchange area during the object program execution interval. At the end of the execution interval the object program exchange package is returned to SCM and the monitor program is continued.

The execution interval for an object program exchange package may be terminated by an input/output interrupt, or by an error condition In these cases the exchange package for the object program is temporarily stored in a SCM area other than its normal location. The exchange package for the interrupting program will be returned to this location and the object program resumed when the interrupt requirement has been satisfied.

Organization of hardware for the exchange sequence is illustrated in figure 2-13 on the following page. Address flow is illustrated in the upper portion of this figure, and control organization in the lower portion. The data format for the information in the exchange package is described in part one of this manual. This format is illustrated in figure 1-3 on page 1-21. Data format will not be discussed in this section of the manual. The following information will deal with the hardware mechanism for implementing the interchange of the data between SCM and the operating registers.



exchange address



exchange control

Fig. 2-13 Exchange Sequence

Exchange address

There are several modes of initiating an exchange sequence and several sources for the exchange package initial address. The normal termination for an exchange package execution interval is caused by an exchange exit instruction in the associated program. In this case the exit mode flag in the PSD register determines the source of the exchange sequence initial address. Abnormal terminations for an exchange package execution interval are caused by I/O section interrupt requests or by error condition flags in the PSD register. The four combinations of situations are individually described below.

Exit instruction - Exit mode flag:
These conditions normally occur in the termination of a monitor program execution interval. In this case the exchange sequence address is formed by adding (Bj) + K + (RAS). The quantity (Bj) + K is formed in a static network as illustrated in figure 2-13. This addition is in an 18 bit ones complement mode. The lowest order 16 bits of this result are transmitted to the XJA register. Concurrently the RAS input path is selected for the second input to the XJA register. The absolute SCM address for the exchange sequence is formed in a static network following the XJA register. This arithmetic is performed in a positive integer mode with the lowest order 16 bits delivered to the BAK register for execution of the exchange sequence. An overflow of the 16 bit length in this addition process is an error condition. This error condition is not sensed in the hardware. Should a monitor program erroneously execute an exchange exit instruction with this condition, the exchange sequence will begin at the SCM address corresponding to the lowest order 16 bits of this sum.

Exit instruction - No exit mode flag:
These conditions normally occur in the termination of an object program execution interval. In this case data transmission from the (Bj) + K static network to the XJA register is blocked. All zero bits enter the XJA register in this position. The NEA input path is selected for the second input to the XJA register. The resulting positive integer sum delivered to the BAK register is simply (NEA).

I/O section interrupt:
This condition blocks the (Bj) + K network data transmission to the XJA register. All zero bits enter the XJA register in this position. The I/O exchange address is transmitted from the I/O section of the CPU to the XJA register for the second input. The resulting positive integer sum delivered to the BAK register is simply the I/O section exchange address.

Error interrupt:
This condition blocks the data transmission from the (Bj) + K static network to the XJA register. All zero bits enter the XJA register in this position. The EEA input path is selected for the second input to the XJA register. The resulting positive integer sum delivered to the BAK register is simply (EEA).

The BAK register and its associated static networks form the mechanism for counting through the 16 address values required in the exchange sequence. This mechanism is also used during the execution of a LCM block copy instruction. A static network at the input to the BAK register selects the proper source for the initial address. In this case the initial address for the exchange sequence enters the BAK register. The address value is increased by one count as each address is delivered to the SAS during the execution of the exchange sequence. This counting process is controlled by the SCM access control unit.

XJA register

The XJA register is a 32 bit clear/enter type register with gated clock pulse control. The register is arranged in two groups of 16 bits each. These two 16 bit quantities determine the absolute address for the exchange sequence. The data in the register is cleared, and new data entered at the beginning of each exchange sequence. This occurs at the end of a clock period in which the following condition exists:

Condition: I/O exchange flag
or: Error exchange flag
or: PXF

The data entered in the XJA register is determined by four control conditions. These four conditions are mutually exclusive. A pair of 16 bit addresses is entered in the XJA register for each of the four control conditions. These pairs are identified below for each condition.

(Bj) + K, (RAS); on condition: PXF & exit mode flag
zero, (NEA); on condition: PXF & no exit mode flag
zero, (EEA); on condition: Error exchange flag
zero, I/O exchange address; on condition: I/O exchange flag

Exchange control

The exchange sequence control mechanism is illustrated in the lower portion of figure 2-13 on page 2-73. The exchange sequence begins with the request interrupt flag (RIF). This flag is set by a number of control conditions as described below. The RIF prevents the current program sequence from advancing beyond the current program instruction word. A static network following the RIF determines when the execution of the current program instruction word is completed. At this time the exchange sequence flag (XSF) is set to begin the interchange of data between the operating registers and the SCM. This interchange requires 16 SCM storage references. The exchange sequence counter (XSK) advances one count as each SCM word reference is made. Concurrently, the block address counter (BAK) advances one count to form the SCM address for the SAS. Data is transmitted from the operating registers to the SWS under the control of a static network following the XSK register. The register content is translated by this static network to select one set of operating registers for each value in the XSK. The presence of the XSF causes the i, j, and k designators in the CIW register to count synchronously with the lower three bits of the XSK. These designator values select the proper X, A, and B registers for the exchange data. When the 16 storage references have been made the JCF is set to complete the exchange sequence and begin execution of the new program sequence.

Request interrupt flag (RIF)

The RIF is a bit register with a separate set and separate clear input. This control flag is set whenever a control condition exists which requires an exchange sequence. One of several mode flags may be set at the same time to determine which of the several control conditions caused the interruption. The RIF remains set until the currently executing program has completed the current program instruction word. The RIF is then cleared and the XSF is set to continue the exchange sequence. The conditions for setting and clearing the RIF are listed below:

Set condition: "go exit" & no "dead start"
or: "I/O exchange request" & no "monitor mode" & no "dead start" & no RIF & no XSF & no "go exit"
or: "breakpoint" & no "dead start" & no RIF & no XSF & no "go exit"
or: "step mode" & no "dead start" & no RIF & no XSF & no "go exit"
or: "error" & no "dead start" & no RIF & no XSF & no "go exit"

Clear condition: "registers quiet" & no F1F & g = 0 & h = 0 & RIF

Exchange sequence flag (XSF)

The XSF is a bit register with a separate set and separate clear input. This control flag is set at the same time that the RIF is cleared. This flag remains set during the period of time required to initiate 16 SCM storage references for the exchange sequence. The XSF is then cleared and the JCF is set to begin execution of the new program sequence. The conditions for setting and clearing the XSF are listed below:

Set condition: "registers quiet" & no FIF & g = 0 & h = 0 & RIF

Clear condition: "go issue" & "XSK = 15"

Program exit flag (PXF)

The PXF is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set simultaneously with the setting of the RIF when the cause of the interruption is an exchange exit instruction in the current program sequence. This flag controls the selection of address data for the XJA register and provides the condition for entering the XJA register with the exchange address data. This flag is then cleared in the following clock period. The condition for setting this flag is as follows:

Condition: "go exit" & "registers free"

Error exchange flag

The error exchange flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set simultaneously with the setting of the RIF when the cause of the interruption is an error condition, breakpoint condition, or a step mode condition. This flag controls the selection of address data for the XJA register and provides the condition for entering the XJA register with the exchange address data. This flag is then cleared in the following clock period. The condition for setting this flag is as follows:

Condition: "breakpoint" & no RIF & no XSF & no "go exit" & no "I/O exchange request"
or: "breakpoint" & no RIF & no XSF & no "go exit" & "monitor mode"
or: "error" & no RIF & no XSF & no "go exit" & no "I/O exchange request"
or: "error" & no RIF & no XSF & no "go exit" & "monitor mode"
or: "step mode" & no RIF & no XSF & no "go exit" & no "I/O exchange request"
or: "step mode" & no RIF & no XSF & no "go exit" & "monitor mode"

I/O exchange flag

The I/O exchange flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set simultaneously with the setting of the RIF when the cause of the interruption is an I/O exchange request. This flag controls the selection of address data for the XJA register and provides the condition for entering the XJA register with the exchange address data. This flag is then cleared in the following clock period. The condition for setting this flag is as follows:

Condition: "I/O exchange request" & no "monitor mode" & no RIF & no XSF & no "go exit"

I/O resume flag

The I/O resume flag is a bit register with a separate set and separate clear input. This control flag is set at the same time as the I/O exchange flag. This flag remains set, however, until the exchange sequence actually begins, or until a higher priority interrupt request supersedes this I/O request. If this flag is still set at the time the exchange sequence begins, an "I/O exchange resume" condition is transmitted to the I/O section of the CPU. The conditions for setting and clearing this flag are as follows:

Set condition: "I/O exchange request" & no "monitor mode" & no RIF & no XSF & no "go exit"

Clear condition: "go exit"
or: XSF

"I/O exchange resume" condition

This condition exists when an I/O exchange request has been honored by the exchange sequence control and the exchange sequence has been initiated. This condition exists for only one clock period. This condition is transmitted to the I/O section of the CPU to acknowledge the I/O exchange request. This condition is defined as follows:

Condition: XSF & I/O resume flag

Exchange issue flag (XIF)

The XIF is a clear/enter type bit register which is cleared at the end of every clock period. This flag requests a SCM storage reference for the exchange sequence using an address from the BAK register. This flag is set for a total of 16 clock periods during the execution of an exchange sequence. These 16 clock periods are generally contiguous, but may be scattered by bank conflicts in SCM as a result of previously initiated SCM references. The condition for setting this flag is as follows:

Condition: XSF & " go issue"

Exchange sequence count register (XSK)

The XSK register is a four bit clear/enter type register with gated clock pulse control. This register counts through the 16 storage references in the execution of the exchange sequence. Each of the 16 values in the XSK register is translated to select a particular register combination for data to the SWS. This register is cleared and entered with the next sequential count at the end of each clock period in which the following condition exists:

Condition: XSF & "go issue"

"XSK = 15" condition

This condition exists when the XSK register is set to all one bits. This is the last step in the 16 step exchange sequence. As this last step is executed the XSF is cleared and the XSK is reset to all zero bits. The JCF is set at this same time to begin execution of the new program sequence.

Program status register

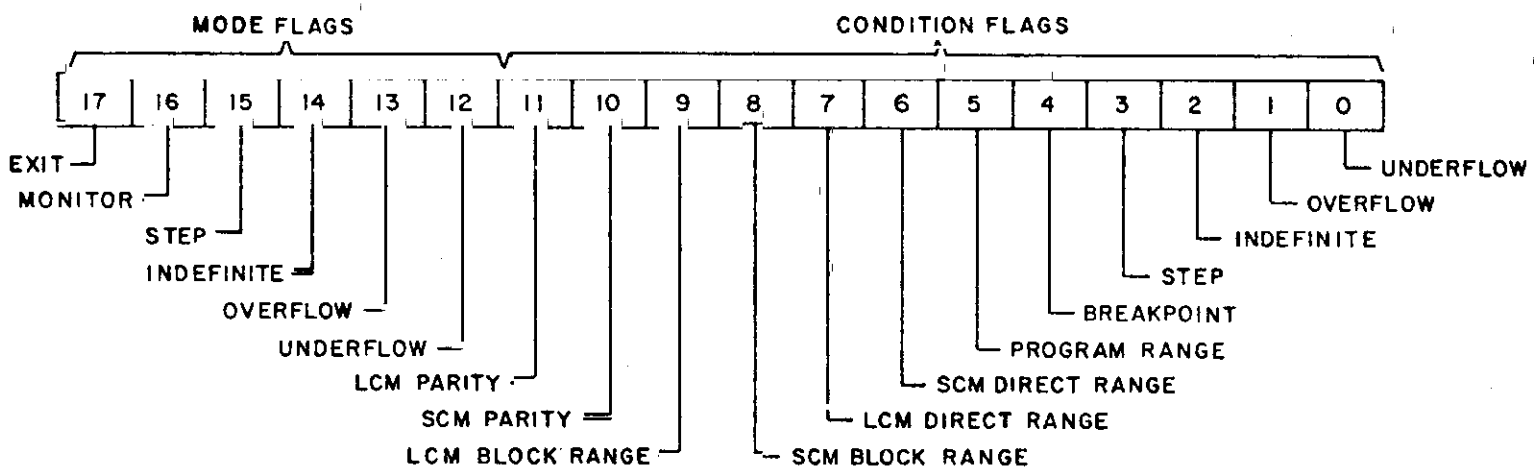
The program status register (PSD) is a collection of individual program status designation flags. There are 18 flags in this register. Six of these flags are mode designators, and 12 are condition designators. The arrangement of these flags in the bit positions in the register is shown in figure 2-14 on the following page.

The PSD register is loaded along with the other CPU registers during an exchange sequence. All 18 bits are entered in the register at this time. The six mode designators remain unaltered throughout the execution interval for the exchange package. The 12 condition designators may be set by transient conditions which occur during the execution interval. All flags are stored in the SCM exchange package at the end of the execution interval.

The execution interval for an exchange package may be terminated by an error condition which occurred during this interval. The monitor program must determine the nature of the error condition by analysing the flags in the PSD portion of the exchange package in SCM. This monitor program may then clear the flag in the exchange package and resume execution of the object program, or it may take some alternate action.

Exit mode flag (PSD bit 17)

The exit mode flag is cleared and entered with data from the exchange package during an exchange sequence. This flag is not altered during the execution interval for the exchange package. This flag controls the source of the exchange package address for the execution of an exchange exit instruction (013). If this flag is set, the exchange package address is (Bj) + K + (RAS). If this flag is not set, the exchange package address is (NEA).



PROGRAM STATUS DESIGNATIONS REGISTER (DSD)

Monitor mode flag (PSD bit 16)

The monitor mode flag is cleared and entered with data from the exchange package during an exchange sequence. This flag is not altered during the execution interval for the exchange package. This flag controls the mode of input/output activity. If this flag is set, the currently executing program can not be interrupted by an I/O interrupt request. If an I/O interrupt request occurs, it will not be honored until the end of the execution interval for the current exchange package.

The monitor mode flag also controls the execution of the reset buffer instructions (0160, 0170). If the monitor mode flag is set, the reset buffer instructions may be executed as described. If the monitor mode flag is not set, a reset buffer instruction is executed as a pass. This provision is to prevent an object program from interfering with I/O activity.

Step mode flag (PSD bit 15)

The step mode flag is cleared and entered with data from the exchange package during an exchange sequence. This flag is not altered during the execution interval for the exchange package. If this flag is set, the current program will be interrupted at the end of each program instruction word. This flag causes the step condition flag to set as the first instruction is issued from the CIW register. The step condition flag then generates an error exit request which terminates the execution interval at the end of the current program instruction word. The terminating exchange package is at absolute address (EEA) in SCM.

Indefinite mode flag (PSD bit 14)

The indefinite mode flag is cleared and entered with data from the exchange package during an exchange sequence. This flag is not altered during the execution interval for the exchange package. This flag enables interruption of the current program on the condition of an indefinite floating point result. The combination of this flag set and the indefinite condition flag set generates an error exit request which terminates the execution interval at the end of the current program instruction word. This program instruction word is not necessarily the word containing the instruction which caused the indefinite condition. Rather, it is the current instruction word at the time the error condition is generated in a functional unit. The terminating exchange package is located at absolute address (EEA) in SCM for this case.

Overflow mode flag (PSD bit 13)

The overflow mode flag is cleared and entered with data from the exchange package during an exchange sequence. This flag is not altered during the execution interval for the exchange package. This flag enables interruption of the current program on the condition of an overflow of the floating point range in a floating point calculation. The combination of this flag set and the overflow condition flag set generates an error exit request which terminates the execution interval at the end of the current program instruction word. This program instruction word is not necessarily the word containing the instruction which caused the overflow condition. Rather, it is the current instruction word at the time the error condition is generated in a functional unit. The terminating exchange package is located at absolute address (EEA) in SCM for this case.

Underflow mode flag (PSD bit 12)

The underflow mode flag is cleared and entered with data from the exchange package during an exchange sequence. This flag is not altered during the execution interval for the exchange package. This flag enables interruption of the current program on the condition of an underflow of the floating point range in a floating point calculation. The combination of this flag set and the underflow condition flag set generates an error exit request which terminates the execution interval at the end of the current program instruction word. This program instruction word is not necessarily the word containing the instruction which caused the underflow condition. Rather, it is the current instruction word at the time the error condition is generated in a functional unit. The terminating exchange package is located at absolute address (EEA) in SCM for this case.

LCM parity condition flag (PSD bit 11)

The LCM parity condition flag is cleared and entered with data from the exchange package during an exchange sequence. In addition, this flag is set whenever a LCM parity error is detected during a LCM read/write cycle. If this flag is set, either from the exchange sequence or from the parity error detection, it generates an error exit request which terminates the execution interval for the exchange package at the end of the current program instruction word. The terminating exchange package is located at absolute address (EEA) in SCM for this case.

SCM parity condition flag (PSD bit 10)

The SCM parity condition flag is cleared and entered with data from the exchange package during an exchange sequence. In addition, this flag is set whenever a SCM parity error is detected during a SCM read/write cycle. If this flag is set, either from the exchange sequence or from the parity error detection, it generates an error exit request which terminates the execution interval for the exchange package at the end of the current program instruction word. The terminating exchange package for this case is located at absolute address (EEA) in SCM.

LCM block range condition flag (PSD bit 09)

This condition flag is cleared and entered with data from the exchange package during an exchange sequence. In addition, this flag is set whenever a block copy instruction is issued from the CIW register which would cause a LCM reference to an address equal to, or greater than, (FLL). The block copy instruction is issued as a pass instruction in this case. If this flag is set, either from the exchange sequence or from the range error detection, it generates an error exit request which terminates the execution interval for the exchange package at the end of the current program instruction word. The terminating exchange package for this case is located at absolute address (EEA) in SCM.

SCM block range condition flag (PSD bit 08)

This condition flag is cleared and entered with data from the exchange package during an exchange sequence. In addition, this flag is set whenever a block copy instruction is issued from the CIW register which would cause a SCM reference to an address equal to, or greater than, (FLS). The block copy instruction is issued as a pass instruction in this case. If this flag is set, either from the exchange sequence or from the range error detection, it generates an error exit request which terminates the execution interval for the exchange package at the end of the current program instruction word. The terminating exchange package for this case is located at absolute address (EEA) in SCM.

LCM direct range condition flag (PSD bit 07)

This condition flag is cleared and entered with data from the exchange package during an exchange sequence. In addition, this flag is set whenever a read LCM (014) or write LCM (015) instruction causes a LCM reference to an address equal to, or greater than, (FLL). Writing into LCM is inhibited in such a case. If this flag is set, either from the exchange sequence or from the range error detection, it generates an error exit request which terminates the execution interval for the exchange package at the end of the current program instruction word. The terminating exchange package for this case is located at absolute address (EEA) in SCM.

SCM direct range condition flag (PSD bit 06)

This condition flag is cleared and entered with data from the exchange package during an exchange sequence. In addition, this flag is set whenever a SCM reference other than a block copy instruction occurs with an address equal to, or greater than, (FLS). Writing into SCM is inhibited in such a case. If this flag is set, either from the exchange sequence or from the range error detection, it generates an error exit request which terminates the execution interval for the exchange package at the end of the current program instruction word. The terminating exchange package for this case is located at absolute address (EEA) in SCM.

Program range condition flag (PSD bit 05)

This condition flag is cleared and entered with data from the exchange package during an exchange sequence. In addition, this flag is set whenever (P) equals zero, (P) equals or exceeds (FLS), or an error exit (00) code is issued from the CIW register. If this flag is set, either from the exchange sequence or from an error detection, it generates an error exit request which terminates the execution interval for the exchange package at the end of the current program instruction word. The terminating exchange package for this case is located at absolute address (EEA) in SCM. The program address (P) is advanced as soon as execution of an instruction word has begun. As a result, this condition flag will set if the last word of the SCM field is used for a program instruction and this word is executed.

Breakpoint condition flag (PSD bit 04)

This condition flag is cleared and entered with data from the exchange package during an exchange sequence. In addition, this flag is set whenever (P) is equal to (BPA). If this flag is set, either from the exchange sequence or from the breakpoint test, it generates an error exit request which terminates the execution interval for the exchange package at the end of the current program instruction word. The terminating exchange package for this case is located at absolute address (EEA) in SCM.

This condition flag normally sets in time to terminate the execution interval before execution of the instruction word located at program address (BPA). In one case, however, it is possible for execution of the instruction word at address (BPA) to begin before this condition flag has taken effect. This case occurs when two increment instructions with 30 bit formats are contained in a single instruction word and both are issued without delays. In this case the error exit request terminates the execution interval for the exchange package at the end of execution of the instruction word located at address (BPA).

Step condition flag (PSD bit 03)

This condition flag is cleared and entered with data from the exchange package during an exchange sequence. In addition, this flag is set whenever the step mode flag is set and an instruction issues from the CIW register. This combination of conditions has the effect of allowing only one instruction word to be executed in this execution interval for the exchange package. If this flag is set, either from the exchange sequence or from the step mode conditions, it generates an error exit request which terminates the execution interval for the exchange package at the end of the current program instruction word. The terminating exchange package for this case is located at absolute address (EEA) in SCM.

Indefinite condition flag (PSD bit 02)

This condition flag is cleared and entered with data from the exchange package during an exchange sequence. In addition, this flag is set whenever an indefinite floating point result is generated in a floating point functional unit. An indefinite result may occur during execution of instructions 30, 31, 32, 33, 34, 35, 40, 41, 42, 44, and 45. If this flag is set, either from the exchange sequence or from an indefinite result, and if the indefinite mode flag is also set, then an error exit request is generated which terminates the execution interval for the exchange package at the end of the current program instruction word. The terminating exchange package for this case is located at absolute address (EEA) in SCM.

Overflow condition flag (PSD bit 01)

This condition flag is cleared and entered with data from the exchange package during an exchange sequence. In addition, this flag is set whenever an overflow of the floating point range occurs in the result from a functional unit. A floating point overflow result may occur in the execution of instructions 30, 31, 32, 33, 34, 35, 40, 41, 42, 44, and 45. If this flag is set, either from the exchange sequence or from an overflow result, and if the overflow mode flag is also set, then an error exit request is generated which terminates the execution interval for the exchange package at the end of the current program instruction word. The terminating exchange package for this case is located at absolute address (EEA) in SCM.

Underflow condition flag (PSD bit 00)

This condition flag is cleared and entered with data from the exchange package during an exchange sequence. In addition, this flag is set whenever an underflow of the floating point range occurs in the result from a functional unit. A floating point underflow result may occur in the execution of instructions 32, 33, 40, 41, 42, 44, and 45. If this flag is set, either from the exchange sequence or from an underflow result, and if the underflow mode flag is also set, then an error exit request is generated which terminates the execution interval for the exchange package at the end of the current program instruction word. The terminating exchange package for this case is located at absolute address (EEA) in SCM.

X registers

The eight X registers are the principal operating registers for the CPU. They are individually designated in this manual by the symbols X0, X1, X2, X3, X4, X5, X6, and X7. These registers are each 60 bits in length and serve as the source and destination for operands in execution of the arithmetic instructions. Each register is a clear/enter type register with gated clock pulse control. Data will remain in an X register until a control condition generated in the X register access control unit specifically gates a clock pulse to clear the data and enter new data. At most one X register can be cleared and entered with new data at the end of any given clock period. The control condition which causes this entry is the "enter register Xd" condition. The selection of the proper X register is specified by a three bit designator (d) originating in the X register access control unit.

Communication between the X registers and the functional units involves a substantial merging of 60 bit data paths and distribution of 60 bit data paths. Almost every functional unit has at least one data path to the X registers and one data path from the X registers. Several of the floating point units have multiple 60 bit data paths. This merging and distribution function is performed in 60 bit static networks preceding and following the X registers themselves. This is illustrated in figure 2-15 on the following page.

Data flow from the functional units to the X registers is treated in two groups. One group, consisting of the multiply, divide, shift, and normalize units, is treated in a static merge network as shown in figure 2-15. Data from these units flows through a complement control network before entering the X registers. The second group, consisting of the remaining functional units and the storage units, merges with the data from the complement control network in a second static merge network. The 60 bits of data from this last network are delivered to all eight of the X registers. The data is entered in one of these registers only when the "enter register Xd" condition is present.

Data flow from the X registers to other parts of the system is treated in a static distribution network following the X registers This network receives 60 bits of data from each of the X registers plus nine bits of control information from the CIW register. This control information corresponds to the i, j, and k designators in the CIW register. Data paths from this distribution network to other parts of the system are divided into three groups. Each group of data paths carries information from a single X register

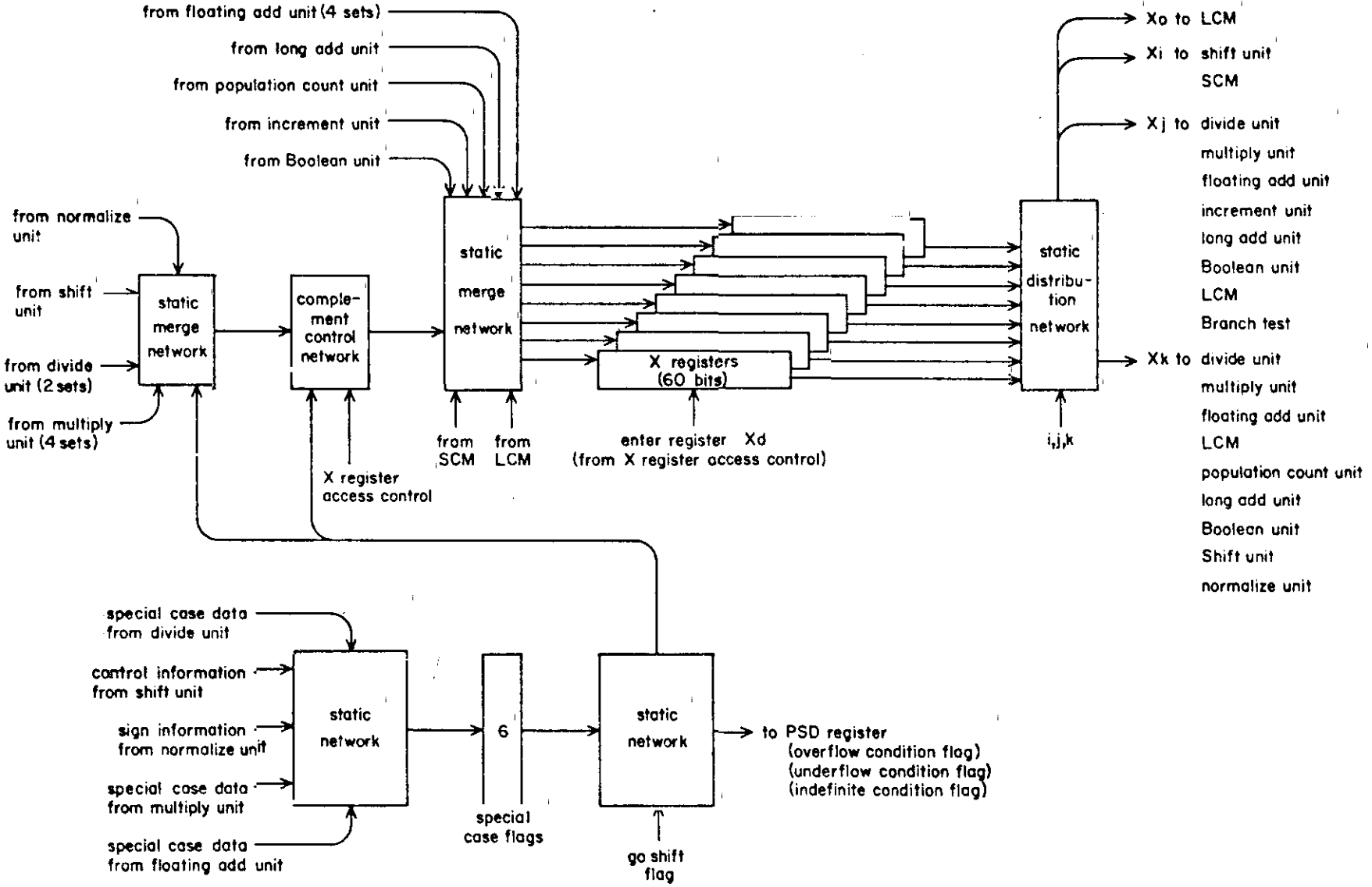


Fig.2-15 X registers

in a given clock period. The X register for each group is specified by the i, j, or k designator in the CIW register. The destinations for each of these groups are listed in figure 2-15. In addition to these three groups, there is a single data path from the X0 register to the LCM. No selection is involved in this data path.

X register complement control

That portion of the X register control which treats special floating point cases and complement of results is illustrated in figure 2-15. A static network merges the special case and sign treatment information from the functional units into six special case flags. These flags are clear/enter type bit registers which are cleared at the end of every clock period. The control information in these flags plus the go shift flag is resolved in a static network to determine the mode of operation in the X register merge networks.

The special case flags in the X register complement control are not named. Two of these flags contain sign information for the shift and normalize functions. One flag contains sign information for the multiply and divide functions. The other three flags hold the special case information for overflow, underflow, and indefinite results. The go shift flag information enters the static network following the special case flags. This is necessary because of the two clock period timing for the shift instruction. The other functional units involved have execution intervals longer than two clock periods and the go unit information is included in the special case flags.

Data arriving at the X registers from functional units in the first merge group is complemented in the complement control network when the result from the functional unit should be negative. The mode of operation of the complement control network is determined by the special case flags. The complement condition for each functional unit involved is described in the corresponding section of this part of the manual. In addition to this complement control function, the special case flags enter the bit patterns for the special floating point formats in the first static merge network when required by the functional units. In these cases the functional units do not transmit a result to the X registers directly. The complement control network operates in a normal manner in these cases to complement the special floating point format if required for a negative result.

The X register complement control transmits the overflow, underflow, or indefinite case information to the PSD register to set the corresponding condition flag. This transmission occurs in the same clock period that the data arrives at the X register.

X register access control

The access control mechanism for the X registers is illustrated in figure 2-16 on the following page. These circuits determine the timing and the register selection for each word entered in an X register. In addition, this unit releases the X register reservation flags at the proper time to correspond with information arrival at the X registers.

The X register access control contains four clear/enter type registers, each of four bit length. These registers are cleared and entered with new data at the end of every clock period. They are individually designated as register #1, register #2, register #3, and register #4 in the X register access control unit. Information flows from register #4 through the other three registers at a rate of one register each clock period. The group of registers constitutes a delay mechanism for the destination X register specification in an instruction execution.

Data enters the four registers through static selection networks. Such a network precedes each register. A delay of zero to four clock periods may be obtained from these circuits, depending on which static network is entered with the data. Each static network contains a data switch which chooses between the data from the preceding register in the chain or new data from outside of the X register access control unit. This data switch is controlled by a functional unit flag for the unit with the corresponding execution time. Data flows from register to register unless the controlling functional unit flag is set. If the flag is set the data relative to that functional unit is entered in the static network and processed down the chain of registers.

No two functional units may deliver data to an X register in the same clock period. The responsibility for avoiding conflicts of this type rests with the instruction issue control rather than the X register access control unit. An instruction is not allowed to issue if the data to the destination X register will conflict with data from another functional unit which is already in process. As a result, the functional unit control of the static network switches in the access control chain will never discard useful data from a higher order register in the chain.

The functional unit with the longest execution time is the divide unit. This unit is not segmented to the degree of the other units, and essentially only one divide operation may be in process at one time. When the divide instruction issues from the CIW register, the i designator is captured in a three bit register in the X register

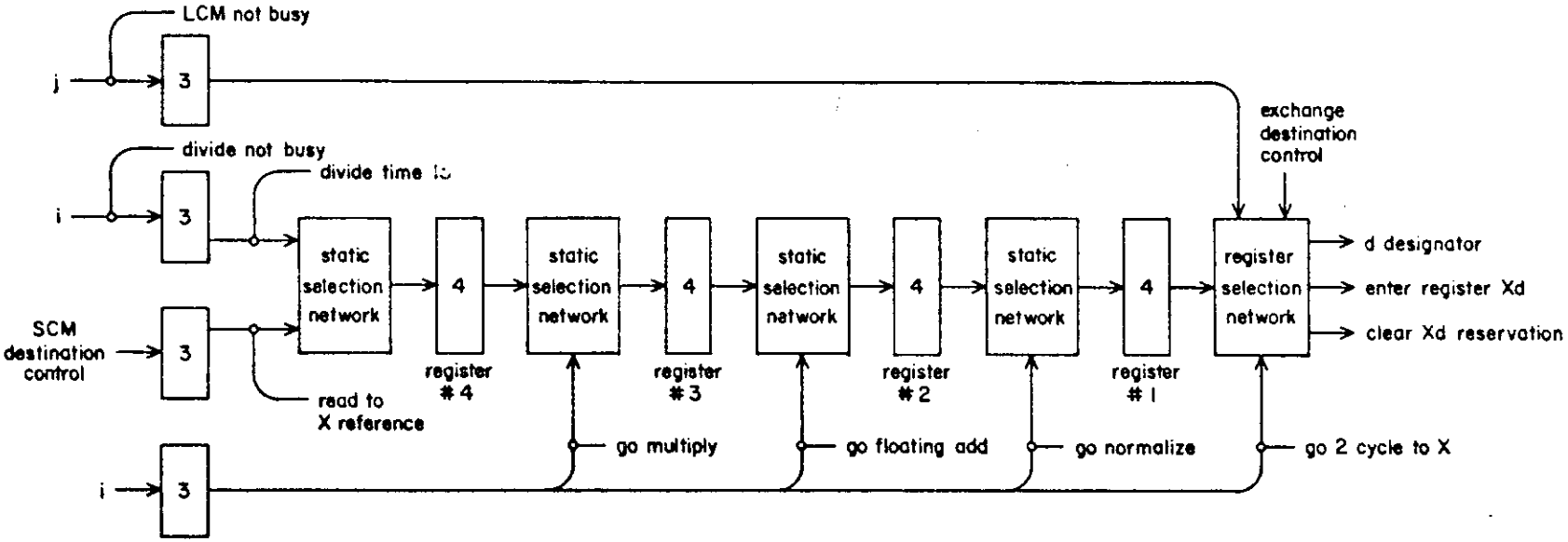


Fig. 2-16 X Register Access Control

access control unit. This register is a clear/enter type register with gated clock pulse control. The register is cleared and entered with the current value of the i designator only when the divide busy flag is cleared. This register receives a copy of the i designator from the CIW register at the end of every clock period until the divide busy flag is set. This designator value is then held in this three bit register until later in the divide sequence. This designator value specifies the X register to which the quotient is to be delivered.

At divide time 15 in the divide unit sequence the data in the three bit register is gated into the X register access chain through a static selection network. The data switch in this static network is controlled by the "divide time 15" condition in the divide unit. During the clock period in which this condition exists, the data in the three bit register is transmitted to register #4 along with a one bit mark to indicate that a three bit code is present. These four bits then move from register to register and control the data entry to the X register four clock periods later.

The SCM access time is longer than any functional unit execution time other than the divide unit. A SCM reference which results in reading a word into a destination X register must be treated in a manner similar to that for a functional unit. Conflicts between a SCM reference and a previously issued divide instruction are prevented in the SCM access control. A SCM reference is delayed if such a conflict would occur. A three bit code is transmitted from the SCM destination control to the X register access control each clock period. This code is entered in a three bit register as shown in figure 2-16. This register is a clear/enter type register which is cleared at the end of every clock period.

A "read to X reference" condition is transmitted from the SCM destination control to the X register access control when the X register destination code is present in the three bit register. This condition causes the static network preceding register #4 to gate the three bit code along with a one bit mark into register #4. These four bits then move from register to register and control the data entry into the proper X register four clock periods later.

Most of the entries to the access control chain require a delayed i designator value. This function is performed by a three bit register in the X register access control unit. This is a clear/enter type register which is cleared and entered with the i designator value at the end of every clock period.

A multiply instruction requires five clock periods for execution. The go multiply flag is set during the second of these clock periods, and the data is transmitted from the multiply unit to the destination X register in the last clock period. The go multiply flag controls the static network data switch between register #4 and register #3 in the X register access chain. This switch is thrown during the second clock period of the multiply instruction execution. At the end of this clock period the delayed i designator value is entered in register #3 along with a one bit mark. Three clock periods later this code will control the entry of the multiply result into the appropriate X register.

The floating add instructions require four clock periods for execution. The go floating add flag controls the data switch in the static network between register #3 and register #2 in the access chain. The go floating add flag is set during the second clock period of a floating add instruction execution. At the end of this clock period the delayed i designator value is entered in register #2 along with a one bit mark. Two clock periods later this code will control the entry of the floating add result in the appropriate X register.

The normalize unit requires three clock periods to execute an instruction. The go normalize flag controls the data switch in the static network between register #2 and register #1 in the access chain. The go normalize flag is set during the second clock period of normalize instruction execution. At the end of this clock period the delayed i designator value is entered in register #1 along with a one bit mark. One clock period later this code will control the entry of the normalize unit result in the appropriate X register.

Go two cycle to X flag

This control flag is set whenever an instruction issues from the CIW register which will deliver a result to an X register in the following clock period. This is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set at the end of a clock period in which the following condition exists:

```
Condition:  "go issue" & "registers free" & g = 1
      or:   "go issue" & "registers free" & g = 2 & h = 0,1,2,3,6,7
      or:   "go issue" & "registers free" & g = 3 & h = 6,7
      or:   "go issue" & "registers free" & g = 4 & h = 3,7
      or:   "go issue" & "registers free" & g = 7
```

X register selection network

The register selection network in the X register access control unit is a static network which directly controls the clock pulse gates for each of the eight registers. This network includes a data switch to select from one of four possible register designator sources. The delayed i designator value is selected whenever the go two cycle to X flag is set. The content of register #1 is selected whenever the mark bit is set in that register. A three bit code from the exchange destination control unit is selected whenever an associated mark bit is transmitted. These three selections should never conflict because of the mutual exclusion of their origins. If none of these three selections are present, a fourth three bit code for the LCM direct access control unit is selected. This three bit code is a delayed j designator value and is used in entering an X register only if an "enter X from LCM" condition is present.

The three bit register for LCM direct access is contained in the X register access control unit. This register is a clear/enter type register with gated clock pulse control. The register is cleared and entered with the j designator value from the CIW register at the end of every clock period in which the LCM busy flag is cleared. When the LCM busy flag is set this code is held in the three bit register until the "enter X from LCM" condition occurs. The three bit code is then used to select the proper X register for the data transmission from LCM.

The register selection network performs two basic functions. It generates the "enter register Xd" condition which gates a clock pulse to clear/enter an X register. It also generates the "clear Xd reservation" condition which is used in the instruction issue control unit to clear the proper X register reservation flag. (See page 2-16.)

"d" designator

The d designator is the three bit code transmitted from the X register selection network to the X registers and to the X register reservation flags to specify the proper register or flag when the "enter register Xd" or "clear Xd reservation" condition is present. This designator is present every clock period as a result of one of the four possible switch positions in the X register selection network. It is used only when the appropriate control condition is present.

B registers

The eight B registers are intended primarily for indexing functions in program execution. These registers are 18 bits in length and are individually identified in this manual by the symbols B0, B1, B2, B3, B4, B5, B6, and B7. The B0 register does not physically exist in the hardware. In the execution of instructions this register appears to contain all zero bits. Information stored in the B0 register is, in effect, discarded.

Each B register is a clear/enter type register with gated clock pulse control. Data will remain in a B register until a control condition generated in the B register access control unit specifically gates a clock pulse to clear the data and enter new data. At most one B register can be cleared and entered with new data at the end of any given clock period. The control condition which causes this entry is the "enter register Be" condition. The selection of the proper B register is specified by a three bit designator (e) originating in the B register access control.

Communication between the B registers and other parts of the system involves a merging and a distribution of 18 bit data paths. These functions are performed by static networks preceding and following the actual B registers. The individual source and destination for these data paths is indicated in figure 2-17 on the following page.

There are four sources for data to the B registers as indicated in figure 2-17. These data paths are merged without selection at the B register end of the data paths. Only one of these sources may transmit data in a given clock period. This merging is performed in a static network which then delivers a copy of the merged data to each of the seven physical B registers. The data is entered in a B register only when the control condition "enter register Be" is present.

The data from the B registers is distributed through a second static network. This network receives 18 bits of data from each of the B registers plus nine bits of control information from the CIW register. This control information corresponds to the i, j, and k designators in the CIW register. Data paths from this distribution network to other parts of the system are divided into three groups. Each group of data paths carries information from a single B register in any given clock period. The B register for each group is specified by the i, j, or k designator appropriate for that group. The destinations for each of these groups are listed in figure 2-17.

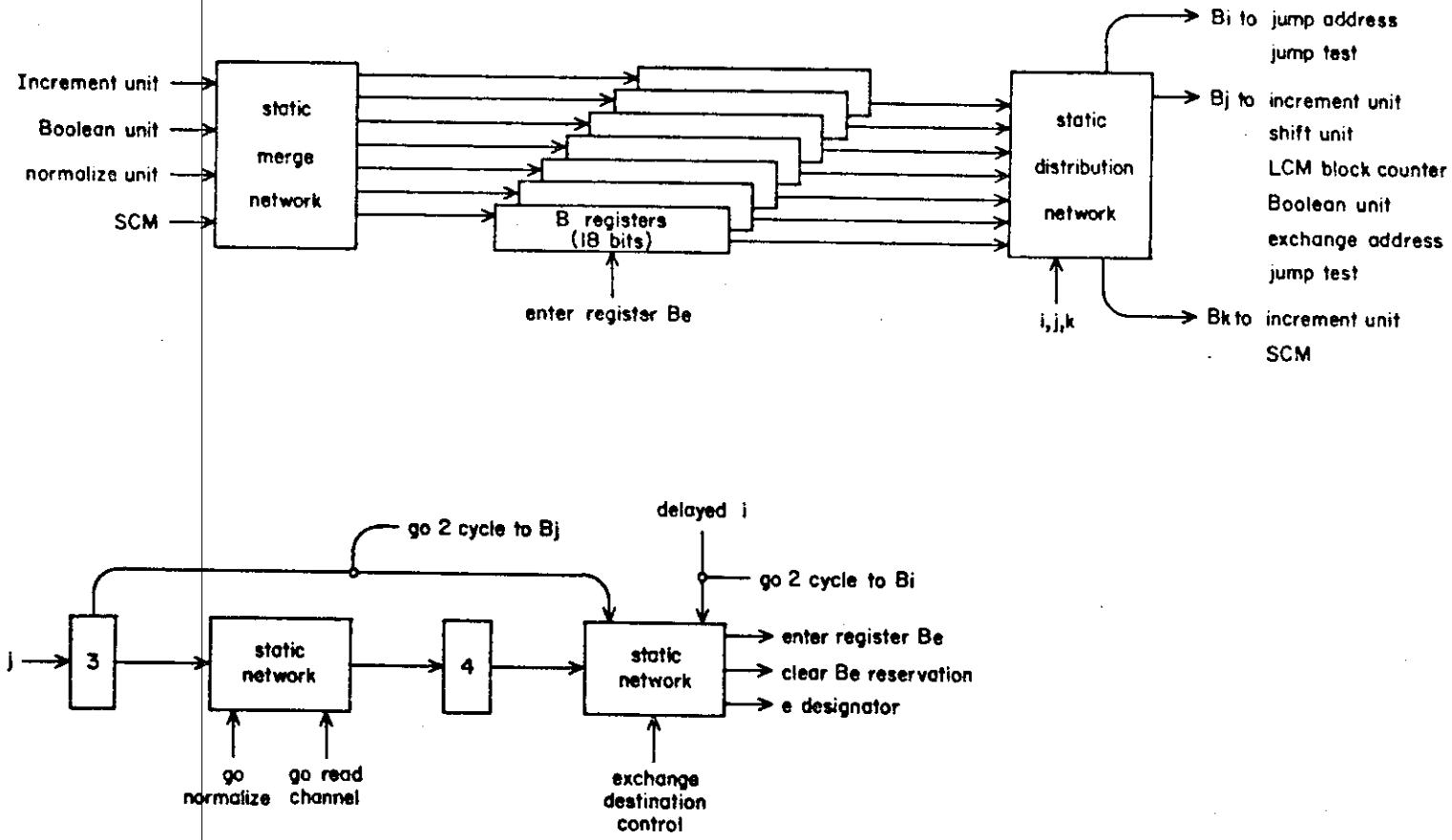


Fig. 2-17 B Registers

B register access control

The access control mechanism for the B registers is illustrated in figure 2-17. These circuits determine the timing and the register selection for each word entered in a B register. In addition, this unit releases the B register reservation flags at the proper time to correspond with information arrival at the B registers.

The B register access control consists of two small registers and two static networks. The first register provides a delayed j designator value for use in register selection. This is a three bit clear/enter type register which is cleared at the end of every clock period. The j designator value is received every clock period, and this value is delayed by the register for use during the following clock period.

The normalize unit is the only functional unit with a three clock period execution time which delivers results to a B register. The go normalize flag is set during the second clock period of a normalize instruction execution. This flag enters the static network in the B register access control and gates the delayed j designator value into the second register along with a one bit mark to indicate that a valid code is present. The gating function is inhibited if the delayed j designator value is zero. In this case the destination B register is B0, and the data is to be discarded.

The 016 instruction causes data to be read from an I/O section channel address register to a B register. This instruction requires three clock periods for execution. The go read channel flag is set during the second clock period of instruction execution. This flag enters the static network in the B register access control in the same manner that the go normalize flag does. The data is gated from the delayed j designator register to the second register along with a one bit mark unless the delayed j designator value is zero.

The second register in the B register access control unit is a four bit clear/enter type register which is cleared at the end of every clock period. This register holds a three bit register designation code plus a one bit mark to indicate a valid reference.

The second static network in the B register access control unit directly controls the clock pulse gates for each of the seven B registers. This network includes a data switch to select from one of four possible register designator sources. These designator sources should never conflict because of the mutual exclusion of their origins. The second register value is selected whenever the mark bit is set in that register. The delayed j designator value is selected whenever the go two cycle to Bj flag is set. The delayed

i designator value is selected whenever the go two cycle to Bi flag is set. A three bit code from the exchange destination control unit is selected whenever an associated mark bit is transmitted. The delayed i designator value for this switch originates in the X register access control and is transmitted to the B register access control unit every clock period.

The static network performs two basic functions. It generates the "enter register Be" condition which gates a clock pulse to clear/enter a B register. It also generates the "clear Be reservation" condition which is used in the instruction issue control unit to clear the proper B register reservation flag. (See page 2-17.)

"e" designator

The e designator is a three bit code transmitted from the B register access control unit to the B registers and the B register reservation flags. This code specifies the proper register or flag when the "enter register Be" or "clear Be reservation" condition is present. This designator is present during any clock period in which one of the four selection conditions is present in the second static network in the B register access control unit.

Go two cycle to Bi flag

This control flag is set whenever an instruction issues from the CIW register which will deliver a result to B register i in the following clock period. This flag is not set if i = 0. In this case the data is discarded. This is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set at the end of a clock period in which the following condition exists:

Condition: "go issue" & "registers free" & g = 6 & i = 1,2,3,4,5,6,7

Go two cycle to Bj flag

This control flag is set whenever an instruction issues from the CIW register which will deliver a result to B register j in the following clock period. This flag is not set if j = 0. In this case the data is discarded. This is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set at the end of a clock period in which the following condition exists:

Condition: "go issue" & "registers free" & g = 2 & h = 6 &
j = 1,2,3,4,5,6,7

A registers

The eight A registers in the CPU are the vehicle for addressing the SCM for operands. These registers are 18 bits in length and are individually identified in this manual by the symbols A0, A1, A2, A3, A4, A5, A6, and A7. The A0 register is special in that it is not used in SCM addressing. This register is intended for storing a reference address or a limit address for comparison with another A register value. It may also be used for some indexing functions in a manner similar to the B registers.

The registers A1, A2, A3, A4, and A5 are used to address the SCM in reading data from SCM to an X register. A read SCM reference is initiated whenever one of these A registers is the destination in execution of an increment instruction. The data from that SCM address is delivered to the corresponding X register. That is, an increment instruction with a destination of the A2 register causes a SCM reference and a data transmission to the X2 register. Such an increment instruction reserves both the A register and the corresponding X register. The A register reservation flag is cleared when the increment unit data arrives at the A register. The X register reservation flag is not cleared until the data arrives from SCM at the X register.

The registers A6 and A7 are used to address the SCM in writing data into SCM from an X register. A write SCM reference is initiated whenever one of these A registers is the destination in execution of an increment instruction. The data from the corresponding X register is delivered to the SWS to be written into the SCM address specified in the A register. Such an increment instruction reserves the A register but not the X register. The X register data is copied into the SWS in the same clock period that the increment instruction issues from the CIW register. The X register is then free in the following clock period. The A register reservation flag is cleared when the increment unit data arrives at the A register.

Each A register is a clear/enter type register with gated clock pulse control. Data will remain in an A register until a control condition generated in the A register access control specifically gates a clock pulse to clear the register and enter new data. At most one A register may be cleared and entered with data at the end of any given clock period. The control condition which causes this entry is the "enter register Af" condition. The selection of the proper A register is specified by a three bit designator (f) originating in the A register access control.

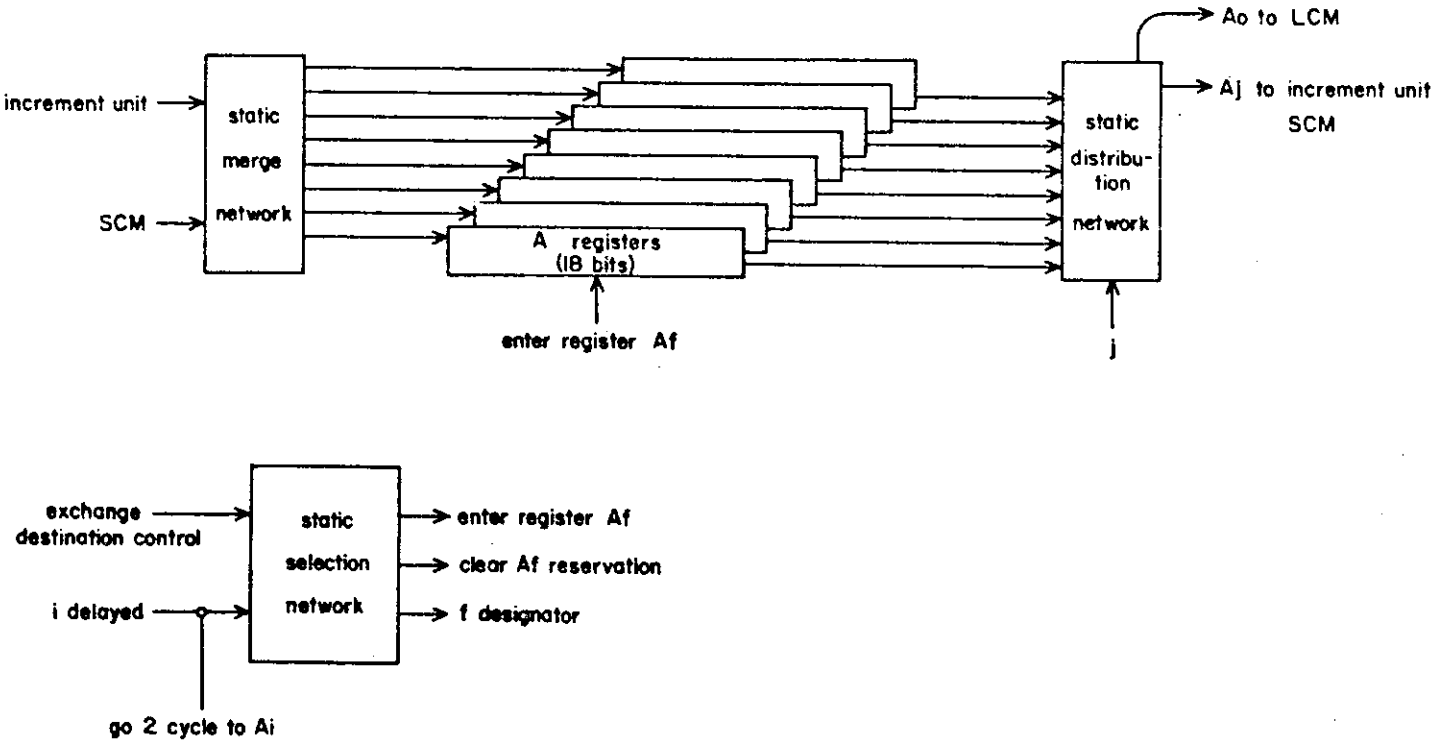


Fig. 2-18 A Registers

Communication between the A registers and other parts of the system is illustrated in figure 2-18 on the preceding page. There are two sources for data to the A registers, the increment unit and SCM. These data paths are merged without selection in a static network prior to the A registers. Only one of these sources may transmit data in a given clock period. The static merge network delivers a copy of the merged 18 bits of data to each of the eight A registers. The data is entered in an A register only when the control condition "enter register Af" is present.

Data from the A registers is distributed through a second static network. This network receives 18 bits of data from each of the A registers plus three bits of control information from the j designator portion of the CIW register. Data paths from this static distribution network go to three destination units. The data in the A0 register is delivered to the LCM access control each clock period. The data in the A register specified by the j designator is delivered to the increment unit and the SCM each clock period. The data path from this static network to the SCM is not used for transmitting a SCM reference address. This data path is used for storing the A register data in the exchange package in SCM during an exchange sequence. The address data for SCM is transmitted directly from the increment unit.

Go two cycle to Ai flag

This control flag is set whenever an instruction issues from the CIW register which will deliver a result to A register i in the following clock period. This is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set at the end of a clock period in which the following condition exists:

Condition: "go issue" & "registers free" & g = 5

"f" designator

The f designator is a three bit code transmitted from the A register access control to the A registers and to the A register reservation flags. This code specifies the proper register or flag when the "enter register Af" or "clear Af reservation" condition is present.

A register access control

The access control mechanism for the A registers is illustrated in figure 2-18. This function is performed in a single static network. This network receives three bits of delayed i designator value from the register in the X register access control unit used for a similar function. A data switch in the static network selects the delayed i designator information only when the go two cycle to Ai flag is set. When this flag is set the static network generates the "enter register Af" condition and the "clear Af reservation" condition. The f designator in this case is set equal to the delayed i designator value.

The exchange destination control unit transmits three bit code plus an associated mark bit to the A register access control during an exchange sequence. The static network in the A register access control selects this three bit code when the associated mark bit is present and transmits this value for the f designator. The "enter register Af" and "clear Af reservation" conditions are also generated in this situation. This sequences the data from SCM into the A registers during an exchange sequence.

Supporting registers

The CPU contains a number of registers which support the operating registers in the execution of programs. These registers are loaded with new information during the execution of an exchange sequence. The information is not altered during the execution interval for an exchange package. These registers are listed below with a description of the individual function performed.

RAS register

This register holds the reference address for SCM during the execution interval for each exchange package. It is an 18 bit clear/enter type register with gated clock pulse control. The data in the RAS register is cleared, and new data entered, only during the exchange sequence. The old data in the RAS register is stored in the terminating exchange package, and the new data is entered from the originating exchange package. The content of the RAS register is used as a reference address in most SCM references from the CPU. Absolute SCM addresses are formed by adding (RAS) to the relative address specified in the CPU instructions. SCM references for instruction fetch or for operands are relative to address (RAS). SCM references from the I/O section of the CPU are absolute addresses and do not use the RAS register.

FLS register

This register holds the field length for SCM during the execution interval for each exchange package. It is an 18 bit clear/enter type register with gated clock pulse control. The data in the FLS register is cleared, and new data entered, only during the exchange sequence. The old data in the FLS register is stored in the terminating exchange package, and the new data is entered from the originating exchange package. The content of the FLS register is used as a limit address for relative SCM references, both for instruction fetch and for operand reference. Any SCM relative address which is equal to, or greater than, (FLS) causes an error exit request which terminates the execution interval for the exchange package.

RAL register

This register holds the reference address for LCM during the execution interval for each exchange package. It is a 24 bit clear/enter type register with gated clock pulse control. The data in the RAL register is cleared, and new data is entered, only during the exchange sequence. The old data in the RAL register is stored in the terminating exchange package, and the new data is entered from the originating exchange package. The content of the RAL register is used as a reference address in all LCM references from the CPU. Absolute LCM addresses are formed by adding (RAL) to the relative address specified in the CPU instructions.

FLL register

This register holds the field length for LCM during the execution interval for each exchange package. It is a 24 bit clear/enter type register with gated clock pulse control. The data in the FLL register is cleared, and new data entered, only during the exchange sequence. The old data in the FLL register is stored in the terminating exchange package, and the new data is entered from the originating exchange package. The content of the FLL register is used as a limit address for relative LCM references. All LCM addresses from the computation section of the CPU are compared with (FLL). Any address which is equal to, or greater than, (FLL) causes an error exit request which terminates the execution interval for the exchange package.

NEA register

This register holds the normal exit address for each exchange package during the execution interval for that exchange package. It is a 24 bit clear/enter type register with gated clock pulse control. Only the lowest order 16 bits of this register are significant. The higher order bits are loaded and stored in the exchange package areas but are not used in the execution of programs. The data in the NEA register is cleared, and new data entered, only during the exchange sequence. The old data in the NEA register is stored in the terminating exchange package, and the new data is entered from the originating exchange package. This register is used during the execution of an exchange exit instruction with the exit mode flag cleared. In this case the current program is terminated with an exchange sequence using (NEA) as the absolute SCM address for the exchange package.

EEA register

This register holds the error exit address for each exchange package during the execution interval for that exchange package. It is a 24 bit clear/enter type register with gated clock pulse control. Only the lowest order 16 bits of this register are significant. The higher order bits are loaded and stored in the exchange package areas but are not used in the execution of programs. The data in the EEA register is cleared, and new data entered, only during the exchange sequence. The old data in the EEA register is stored in the terminating exchange package, and the new data is entered from the originating exchange package. This register is used whenever an error exit occurs during the execution interval for an exchange package. In this case (EEA) is the absolute address in SCM for the terminating exchange sequence.

BPA register

This register holds a break point address for each exchange package during the execution interval for that exchange package. It is an 18 bit clear/enter type register with gated clock pulse control. The data in the BPA register is cleared, and new data entered, only during the exchange sequence. The old data in the BPA register is stored in the terminating exchange package, and the new data is entered from the originating exchange package. The content of the BPA register is compared with the content of the P register each clock period. An error exit request is generated whenever these two quantities are identical. This allows a program to be executed to a particular program address and the execution interval terminated for debugging purposes.

Small Core Memory

The description of the small core memory (SCM) hardware is divided into several sections. The overall organization of these sections is illustrated in figure 2-19 on the following page. Addresses arrive at the storage address stack (SAS) from other parts of the CPU. The SCM access control unit determines the priority of SCM requests when two requests occur simultaneously. This unit also controls the entry of addresses in the SAS. When the SAS data backs up because of SCM bank conflicts the SCM access control stops instruction issue until the conflicts have been resolved.

The SAS provides a buffer area for addresses arriving at SCM from other parts of the CPU. A maximum of three addresses may be held in this area when a backup situation occurs. The SCM destination control monitors the addresses leaving the SAS for individual SCM banks. This unit is responsible for determining when a bank conflict would occur and blocking the address in the SAS until the conflict is resolved. The SCM destination control is also responsible for the timing of data leaving the SCM banks for other parts of the CPU.

There are 32 SCM banks in the system. Each bank has its own sequence control and address and operand register. Each bank executes a storage read/write cycle independent of the timing of other SCM banks. Data is transmitted from the SCM bank operand registers to other parts of the CPU over a common data transmission path. There are three destinations for this data as illustrated in figure 2-19. The SRO register is a distribution point for data to all parts of the system other than the instruction stack and the X registers. The instruction stack and X registers receive data directly from the individual bank operand registers.

Data is transmitted from other parts of the CPU to the SCM bank write operand registers via the storage word stack (SWS). This is a stack of seven 60 bit registers which shifts data along from one register to the next until the proper time for entry in the designated SCM bank operand register. This is a delay mechanism to compensate for the read access time in the SCM bank read/write cycle. Shifting in the SWS is controlled by the SCM access control so that the address and the associated data arrive at a SCM bank at the proper time.

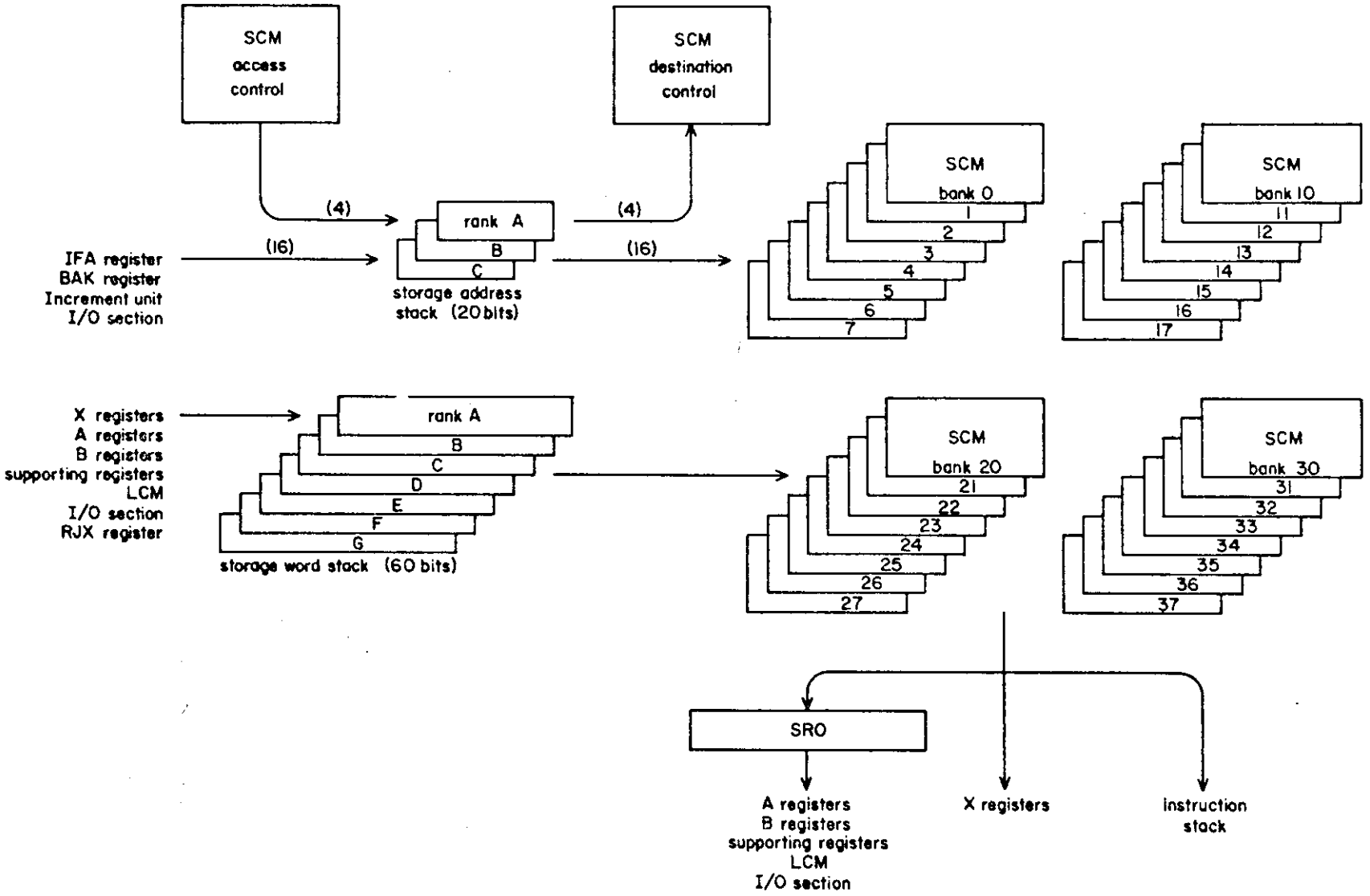


Fig. 2-19 SCM Organization

Storage address stack

The storage address stack (SAS) is a buffer area for addresses arriving at SCM from other parts of the CPU. There are three registers in the SAS, each of 20 bit length. These registers hold a 16 bit SCM address plus a four bit destination code for interpretation by the SCM destination control unit. Each register has an associated control flag which is set when a valid address is present in the register. These parts are illustrated in figure 2-20 on the following page.

Rank A of the SAS is used as a buffer register for incoming addresses when no conflicts are present in the SCM. An address arrives at rank A in any given clock period and leaves for a SCM bank address register in the following clock period. No other register is involved even if a new address arrives every clock period. When a storage bank conflict occurs, the address in rank A is held in that register until the conflict is resolved. This may be for one clock period, or for as many as nine clock periods. Should another address arrive at the SAS during this period it will be routed to rank B. When this occurs the SCM access control unit signals the instruction issue control to stop issuing new instructions until the backup condition has been resolved. There may be one address in process in the increment unit at this time which cannot be stopped. Should this be the case, the address from the increment unit will arrive at the SAS in the following clock period and will be routed to rank C. No further addresses will be accepted by the SAS until the backlog has been cleared up.

When a storage bank conflict has caused a backup situation in the SAS the addresses must leave the SAS in the same order in which they arrived. The address in rank B or the address in rank C cannot leave the SAS until the address in rank A has been transmitted. When rank A has been cleared the address in rank B must follow next, and finally rank C. When the address in rank C has been transmitted the SAS is ready to receive new addresses. These will be routed to rank A.

There are four sources for addresses to the SCM as illustrated in figure 2-20. Two of these sources are used for multiple functions. There are six basic functions which require a SCM reference. The SCM access control unit determines the priority when two or more of these functions request a SCM reference in the same clock period. Only one address may be accepted in a given clock period, and the choice is dictated by the following priority table.

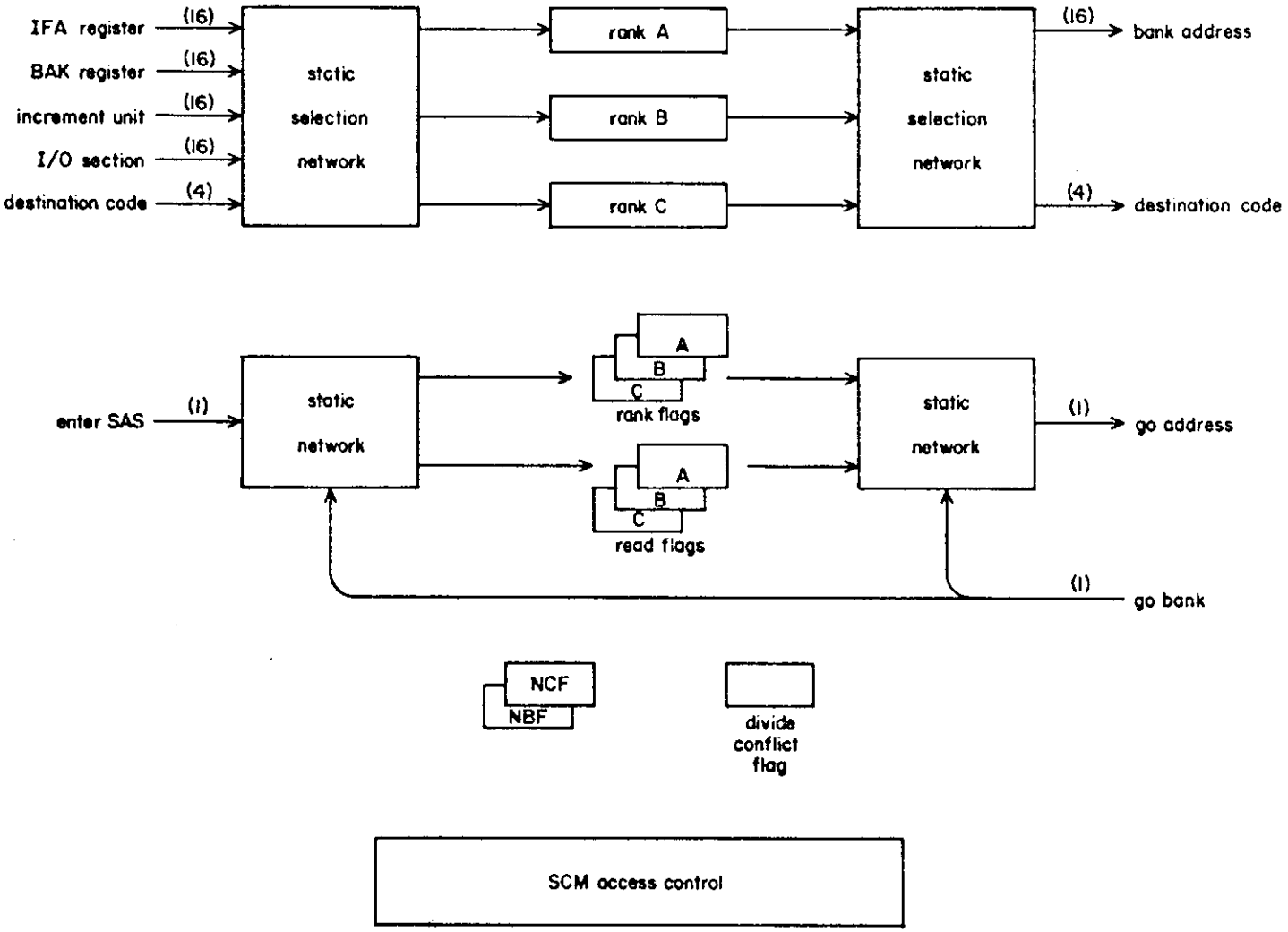


Fig. 2-20 Storage Address Stack (SAS)

First priority - Exchange sequence request
Second priority - Increment unit request
Third priority - Return jump exit request
Fourth priority - I/O section request
Fifth priority - Instruction fetch request
Sixth priority - LCM block copy request

Some of the requests in the above table cannot conflict because of exclusion in the source of the requests. For example, an exchange sequence request cannot occur in the same clock period with an increment unit request because all instructions must be completed before the exchange sequence begins. The table illustrates the manner in which the hardware decides the priority where conflicts can occur.

The SCM access control unit selects the data transmission paths from the four source registers to the SAS. These paths are selected by static control conditions as described below.

Enter an address from the increment unit:
Condition: go increment to storage flag

Enter an address from the I/O section:
Condition: IOF & NBF & no SXF & no go increment to storage flag

Enter an address from the IFA register:
Condition: F1F & NBF & no IOF & no go increment to storage flag
or: SXF & NBF

Enter an address from the BAK register:
Condition: NBF & no IOF & "go block copy'
or: XIF

"Enter SAS" condition

This condition is present whenever an address is being transmitted to the SAS. It is generated in the SCM access control unit and is essentially a merging of the four specific data transmission conditions listed above. This condition is present in the following cases:

Condition: go increment to storage flag
or: IOF & NBF
or: SXF & NBF
or: F1F & NBF
or: "go block copy" & NBF
or: XIF

SCM address tags

Each address which enters the SAS has associated with it a four bit destination code. This code is interpreted by the SCM destination control unit when the address leaves the SAS for a SCM bank. The code is generated in the SCM access control unit as a result of interpreting the individual access requests from other parts of the CPU. The SCM access control unit makes use of a delayed i designator value in forming the four bit destination code. This value is transmitted every clock period from a register in the X register access control unit to the SCM access control unit. The table below lists the octal tag values which are encoded for each of the requesting functions.

00 - not used	10 - not used
01 - read to instruction stack	11 - read to X1 register
02 - exchange sequence	12 - read to X2 register
03 - return jump exit	13 - read to X3 register
04 - read to LCM	14 - read to X4 register
05 - read to I/O section	15 - read to X5 register
06 - write from LCM	16 - write from X6 register
07 - write from I/O section	17 - write from X7 register

The information in the above table may be more precisely expressed in terms of the logical conditions for each bit of the four bit tag. The conditions expressed below are those which the hardware uses in generating each of the tag bits.

Bit 0 of the address tag is present on:
Condition: FlF & NBF & no IOF & no go increment to storage flag
or: SXF & NBF
or: IOF & NBF & no SXF & no go increment to storage flag
or: go increment to storage flag & delayed i designator = 1,3,5,7

Bit 1 of the address tag is present on:
Condition: go increment to storage flag & delayed i designator = 2,3,6,7
or: NBF & no IOF & "go block copy" & "LCM block write"
or: IOF & no SXF & NBF & no go increment to storage flag & "I/O write"
or: SXF & NBF
or: XIF

Bit 2 of the address tag is present on:
Condition: NBF & no IOF & "go block copy"
or: go increment to storage flag & delayed i designator = 4,5,6,7
or: IOF & NBF & no SXF & no go increment to storage flag

Bit 3 of the address tag is present on:
Condition: go increment to storage flag

"Advance IFA" condition

This condition is generated in the SCM access control unit when an address is entered in the SAS from the IFA register. This condition causes the content of the IFA register to be increased by one count at the end of the clock period in which the condition exists.

Condition: F1F & NBF & no IOF & no go increment to storage flag

"Advance BAK" condition

This condition is generated in the SCM access control unit when an address is entered in the SAS from the BAK register. This condition causes the content of the BAK register to be increased by one count at the end of the clock period in which the condition exists.

Condition: NBF & no IOF & "go block copy"
or: XIF

"Reduce LCM block count" condition

This condition is generated in the SCM access control unit when an address is entered in the SAS for a LCM block copy instruction. This condition causes the content of the LCM block counter register to be decreased by one count at the end of the clock period in which the condition exists.

Condition: NBF & no IOF & "go block copy"

"Accept I/O" condition

This condition is generated in the SCM access control unit when an address is entered in the SAS from the I/O section of the CPU. This condition releases the I/O section address mechanism and allows it to advance to the next channel requirement.

Condition: IOF & NBF & no SXF & no go increment to storage flag

SAS rank A register

This is a 20 bit clear/enter type register with gated clock pulse control. This register is used as the principal buffer register in the SAS. It is the only rank of the SAS which is used when the SCM is free of bank conflicts. This register is cleared, and new data entered, at the end of a clock period in which the following condition is present:

Condition: "go bank" & no rank B flag
or: no rank A flag & no rank B flag

SAS rank B register

This is a 20 bit clear/enter type register with gated clock pulse control. This register is used as a backup for the rank A register. When the rank A register is filled and the address cannot be delivered to a SCM bank address register, this register receives the next incoming address to the SAS. This register is cleared, and new data entered, at the end of a clock period in which the following condition is present:

Condition: rank A flag & no rank B flag & no "go bank"

SAS rank C register

This is a 20 bit clear/enter type register with gated clock pulse control. This is the third and last register in the SAS. It is used as a final backup for the arrival of an address at the SAS when the rank A and rank B registers are both filled. This register is cleared, and new data entered, at the end of a clock period in which the following condition is present:

Condition: rank B flag & no rank C flag

Rank A flag

This flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set whenever the data in the SAS rank A register is a valid address for SCM. The condition for setting this flag is as follows:

Condition: "enter SAS" & "go bank" & no rank B flag
or: "enter SAS" & no rank A flag & no rank B flag
or: rank A flag & no "go bank"

Rank B flag

This flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set whenever the data in the SAS rank B register is a valid address for SCM. The condition for setting this flag is as follows:

Condition: rank A flag & no rank B flag & no "go bank" & "enter SAS"
or: rank A flag & rank B flag
or: rank B flag & no "go bank"

Rank C flag

This flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set whenever the data in the SAS rank C register is a valid address for SCM. The condition for setting this flag is as follows:

Condition: rank B flag & no rank C flag & "enter SAS"
or: rank B flag & rank C flag
or: rank C flag & no "go bank"

Read A flag

This flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set whenever the SAS rank A register data is transmitted to the SCM bank address registers. The condition for setting this flag is as follows:

Not: rank B flag & no rank A flag & no "go bank"
nor: rank A flag & rank B flag & "go bank"
nor: rank C flag & no rank B flag & no "go bank"
nor: rank C flag & rank B flag & no rank A flag & "go bank"

Read B flag

This flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set whenever the SAS rank B register data is transmitted to the SCM bank address registers. The condition for setting this flag is as follows:

Condition: rank B flag & no rank A flag & no "go bank'
or: rank B flag & rank A flag & "go bank"

Read C flag

This flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set whenever the rank C register data is transmitted to the SCM bank address registers. The condition for setting this flag is as follows:

Condition: rank C flag & no rank B flag & no "go bank"
or: rank C flag & rank B flag & no rank A flag & "go bank"

Divide conflict flag

This flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set by a condition in the divide unit at the proper time to block SCM activity which might conflict with the divide result arriving at the destination X register. No test is made to determine if the SCM reference will actually deliver a result to an X register. The condition for setting this flag is:

Condition: "divide time 13"

No conflict flag (NCF)

This flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set at the end of a clock period when the SAS is idle, or is active and is delivering a result to a SCM bank address register. This flag is cleared and left cleared at the end of a clock period in which an address is present in the SAS but has not been transmitted to a SCM bank. The condition for setting this flag is:

Condition: "go bank"
or: no rank A flag & no rank B flag & no rank C flag

No backup flag (NBF)

This flag is a misnomer as it is actually a static condition resulting from two other flags. This flag is normally set. The flag appears to clear when the rank B register receives an address. It appears to remain cleared until the rank B register and rank C register have been cleared. Where this flag is referenced it is the equivalent of the following condition:

Condition: no rank B flag & no rank C flag

"Go address" condition

This condition is generated in the SAS when an address is present in one of the three register ranks and the divide conflict flag is cleared. It does not necessarily assure the delivery of an address to a SCM bank address register. This may be prevented by a busy flag at the designated bank. The "go address" condition is defined as follows:

```
Condition:  rank A flag & no divide conflict flag
      or:   rank B flag & no divide conflict flag
      or:   rank C flag & no divide conflict flag
```

"Go bank" condition

This condition is generated in a SCM bank when the bank is addressed and the bank is idle. Each of the 32 banks has the ability to generate this condition, but only one (the addressed bank) may generate the condition in a given clock period. This condition is defined as follows:

```
Condition:  "go address" & no bank 00 busy flag & "bank 00 addressed"
      or:   "go address" & no bank 01 busy flag & "bank 01 addressed"
      or:   "go address" & no bank 02 busy flag & "bank 02 addressed"
      or:   "go address" & no bank 03 busy flag & "bank 03 addressed"
      ..........................................................
      or:   "go address" & no bank 36 busy flag & "bank 36 addressed"
      or:   "go address" & no bank 37 busy flag & "bank 37 addressed"
```

"Bank 00 addressed" condition
"Bank 01 addressed" condition
.....
"Bank 37 addressed" condition

These conditions refer to the lowest order five bits of the address leaving the SAS for a SCM bank address register. The bank selection is made on the basis of these five bits. If the lowest order five bits have a value 00, the "bank 00 addressed" condition exists. If the lowest order five bits have a value 01, the "bank 01 condition" exists. Etc.

SCM banks

There are 32 SCM banks in the system. Each has its own independent sequence control, address register, and write operand register. Any one bank may begin a read/write cycle in a given clock period and complete that operation nine clock periods later. It is thus possible to have ten SCM banks in various phases of activity in any one clock period. Each of the 32 SCM banks has the same configuration. The remainder of this section describes one of these banks as illustrated in figure 2-21 on the following page.

A SCM bank read/write cycle is initiated by a "go bank xx" condition. This condition is generated in the SCM bank sequence control unit from information transmitted from the SAS and information within the bank sequence control. The "bank xx addressed" condition is determined from the lowest order five bits of the address transmitted from the SAS. Each of the 32 SCM banks essentially recognizes its own code in these lowest order five bits.

"Go bank xx" condition

This condition is a component of the "go bank" condition described in the previous section. Each of the 32 SCM banks generates a "go bank xx" condition when the corresponding bank read/write cycle is initiated. The merging of these 32 conditions constitutes the "go bank" condition. The "go bank xx" condition is present on the following condition:

Condition: "go address" & "bank xx addressed" & no bank xx busy flag

The xx in this expression refers to the specific bank number.

Bank xx busy flag

This flag is the mechanism to lock out further initiations of a bank read/write cycle until a previously initiated cycle is completed. This flag is a bit register with a separate set and separate clear input. The flag is set by a "go bank xx" condition when the bank read/write cycle begins and remains set until the last clock period of that cycle. It is set for a total of nine clock periods.

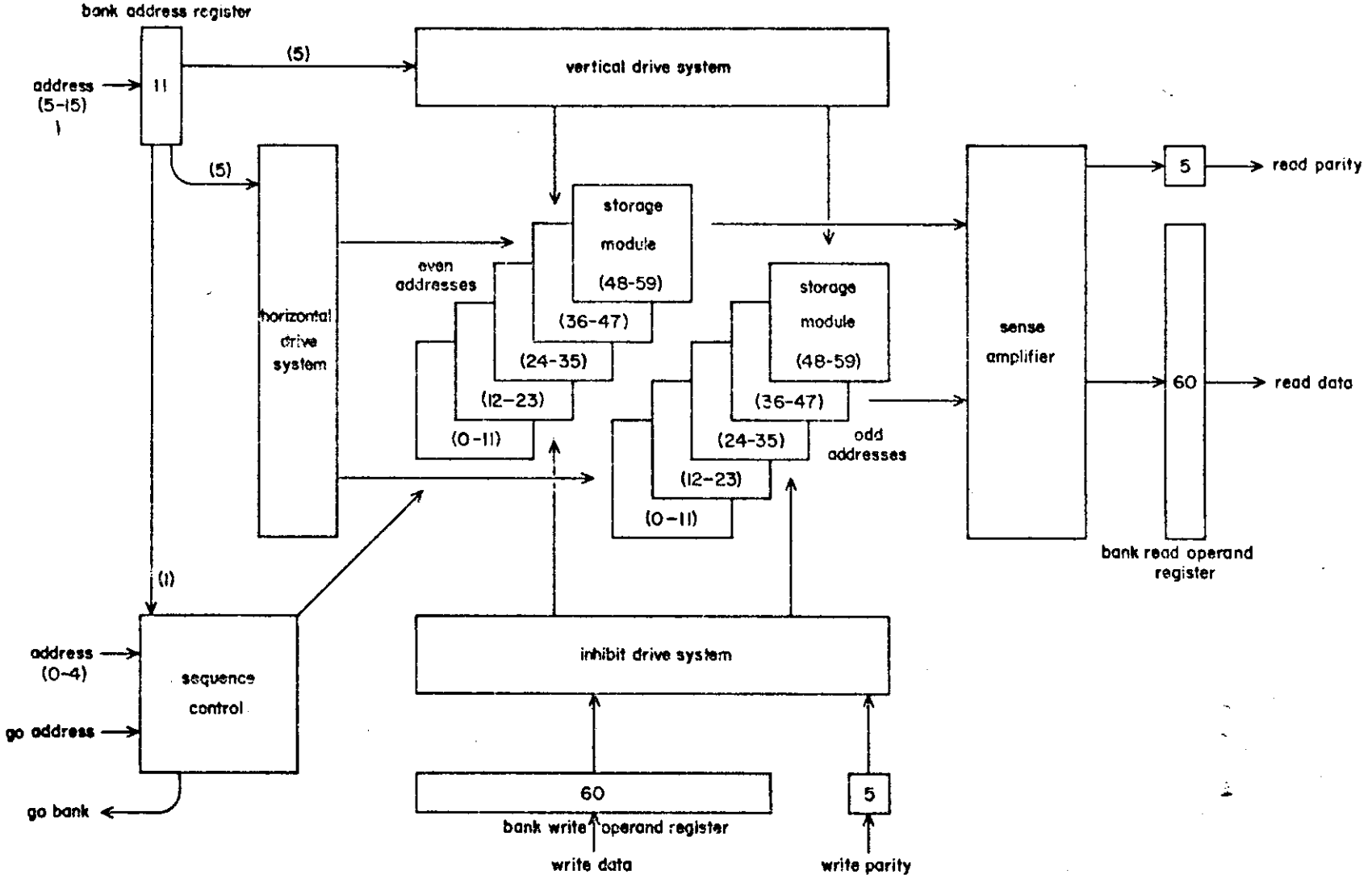


Fig. 2-21 SCM Bank

Bank sequence control

Each SCM bank has its own sequence control unit. This unit forms a sequence of ten conditions which control the activity within the SCM bank in ten consecutive clock periods. The chronological sequence of events in a SCM bank read/write cycle is listed below.

CP00 Transmit address from SAS to SCM bank address register
Go address condition
Bank xx addressed condition
Go bank xx condition
Set bank xx busy flag

CP01 Begin read drive current

CP02

CP03

CP04 End read drive current
Sample sense amplifier output to read operand register

CP05 Transmit read data to distribution network
Transmit write data to write operand register

CP06 Begin write drive current
Begin selected inhibit drive currents

CP07

CP08

CP09 End write drive current
End selected inhibit drive currents
Clear bank xx busy flag

SCM storage modules

Each SCM bank contains ten storage modules. These modules consist of a core memory stack containing 32 x 32 x 13 cores plus the associated supporting circuits. These modules each supply 1024 twelve bit words plus a parity bit for each 12 bit word. The 60 bit SCM word is formed from five storage modules. All of the storage locations with address bit 5 a zero value are contained in one set of

five storage modules. All of the storage locations with address bit 5 a one value are contained in the other set of five storage modules Consecutive addresses in SCM then proceed through the lower set of five storage modules in all 32 banks and then the higher set of five storage modules in all 32 banks. The horizontal and vertical drive system for the storage modules is distributed throughout the ten physical modules. Bits 6 through 10 of the SCM address are translated into the vertical drive selection. Bits 11 through 15 are translated into the horizontal drive selection.

Read operand register

Four SCM banks share a 65 bit read operand register. This register holds 60 bits of data plus five bits of parity information. This is a clear/enter type register which is cleared at the end of every clock period. The data from a given SCM bank resides in this register for only one clock period in the bank read/write cycle. The four SCM banks sharing each read operand register can never conflict in its use because only one SCM bank read/write cycle can be initiated in any given clock period. From a logical standpoint one read operand register would suffice for all 32 SCM banks. There are physically eight of these registers because of the extensive data path problems associated with them.

Data is transmitted to the read operand register from the bank sense amplifiers. There are 65 sense amplifiers in each SCM bank. These amplifiers contain a static network which gates the data to the read operand register during one clock period in the read/write cycle. The control for this gate is in the bank sequence control unit.

Write operand register

Each SCM bank has a 65 bit write operand register. This is a clear/enter type register with gated clock pulse control. The register is cleared and new data entered at the end of CP05 in the bank read/write sequence. This data remains in the register until CP05 in the next read/write cycle in that bank. The data in the write operand register is used during the last four clock periods in the read/write cycle. Data is entered in the core memory stack by selectively inhibiting the coincident current action in individual planes of the stack. The five parity bits are treated in the same manner as the data bits in this process.

Storage Word Stack

The storage word stack (SWS) is a buffer area for 60 bit data words which are to be written into SCM. There are seven ranks of 60 bit registers in this stack. This is substantially more than the number of ranks in the SAS because the SWS must delay the write data by the read access time in the SCM bank read/write cycle. The organization of the SWS is illustrated in figure 2-22 on the following page.

Rank A, rank B, and rank G in the SWS correspond with the three register ranks in the SAS. These three registers have gated clock pulse control and may hold information when a SCM conflict causes a backup condition in the SAS. The last four ranks of the SWS have no clock pulse control and continuously shift data toward the SCM write data paths. These four ranks correspond to the read access time in the SCM bank read/write cycle.

Data arrives at the SWS from a number of other CPU registers as well as the LCM and I/O section. The data from these sources enters the SWS in a number of locations. Those data paths which enter the rank A and rank B registers are supervised by the SCM access control unit. Those data paths which enter the rank E and rank F registers are supervised by the SCM destination control unit. Each of the static networks illustrated in figure 2-22 which has multiple data input paths contains a switch which may be thrown to one of two positions. These switches are thrown by the supervising control unit at the proper time to gate data into the SWS. The data then moves from rank to rank and arrives at the SCM bank write operand register in the proper clock period for controlling the write data.

The static selection network preceding the rank A register in the SWS is more elaborate than the other selection networks. This network includes an eight position selection switch in one branch of the basic two position switch. The result is effectively a nine position selection switch. The normal position for the basic two position switch selects the contents of register Xi. The alternate position of this basic switch has the eight additional selection possibilities. The eight position switch is controlled by the lowest order octal digit in the XSK register. This switch scans the A, B, P, and supporting registers during the execution of an exchange sequence. The timing of the register data transfers in this sequence is listed in part 3 of this manual under the 013 instruction.

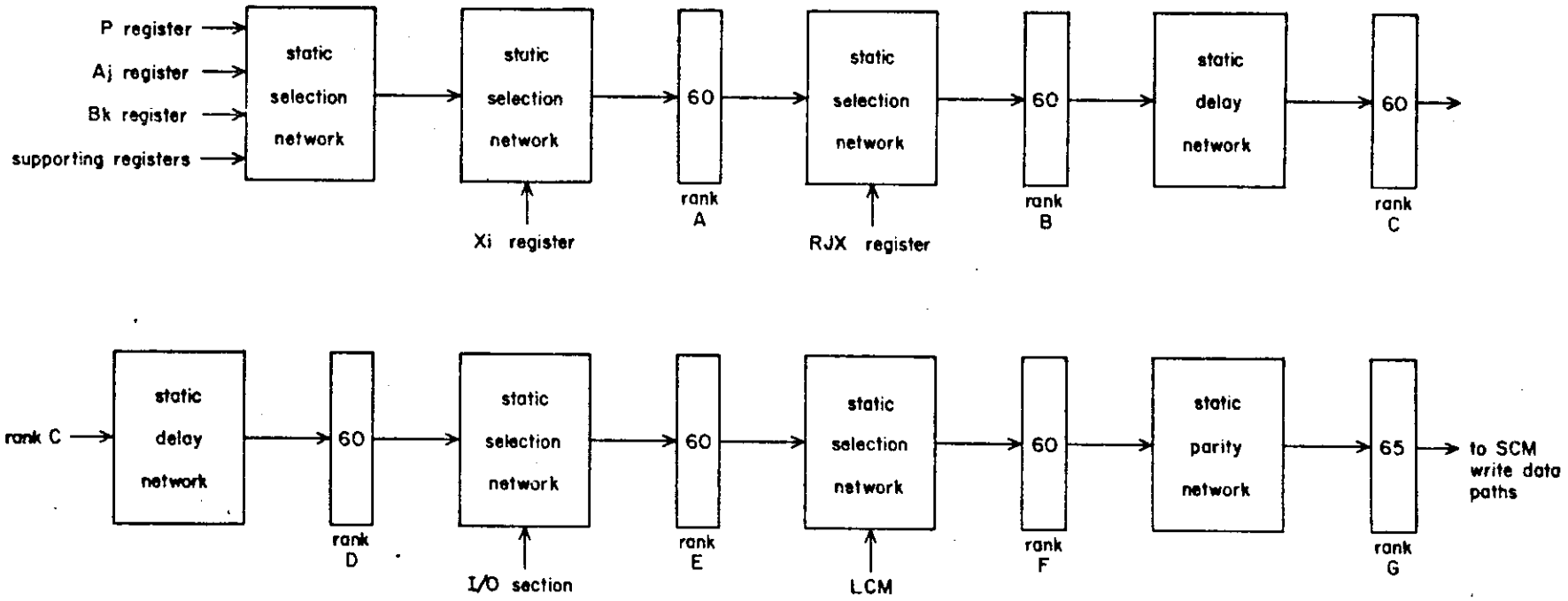


Fig. 2-22 Storage Word Stack

SWS rank A register (SWA)

The SWA register is a 60 bit clear/enter type register with gated clock pulse control. This is the first rank of the SWS. The SWA register is normally cleared and entered with (Xi) at the end of each clock period. This data is then passed along to the rank B register (SWB) in the following clock period. The data is used for writing into SCM only when an increment instruction is executed which requires a SCM store operation. In all other cases the information is shifted through the seven ranks of the SWS and discarded. This normal mode of operation is altered in the event of a SCM conflict, or in the case of executing an exchange sequence.

A SCM conflict occurs when a SCM bank is not able to accept an address from the SAS because the previous read/write cycle has not been completed. In this case the data in the SWA register is held until the conflict has been resolved. The data must move through the SWS in the proper sequence so that the address leaving the SAS and the data word leaving the SWS are in the right time relationship to match the bank read/write cycle. The condition for clearing and entering the SWA register with new data is as follows:

Condition: NCF
or: NBF

"Switch SWA" condition

This condition is present only during the execution of an exchange sequence. It causes the data switch in the static network before the SWA register to switch from the Xi register path to an alternate path. This switch is thrown for the amount of time required to transfer data for the first eight words in the exchange package. These eight words are normally transferred in consecutive clock periods. A delay may occur in this sequence if a SCM bank conflict exists. In this case, the SWA register holds the information until the conflict is resolved in the same manner described above. The data switch returns to its normal position during the time of the last half of the exchange package data transfer. During this interval the X register data is stored in the exchange package. The i, j, and k designators in the CIW register are used to select the specific X, A, and B registers during the exchange sequence. These three octal values are entered in the CIW register by the exchange sequence control unit and have the same value in each clock period of the exchange sequence as is contained in the lowest octal digit of the XSK register.

The scanning of the P register and the supporting registers is accomplished by the eight position switch at the SWA register during the first half of the exchange package data transfer. The i, j, and k designators in the CIW register are not used in this selection. The data from these registers is routed to the uppermost 24 bits of the 60 bit data path to the SWA register. Concurrently, the j and k designators in the CIW register route the appropriate A register and B register values to the lowest order 48 bit positions. A special data path is provided to gate the BPA register data to the position in the exchange package which would normally be occupied by the B0 register value. This selection is performed by the same eight position switch which controls the upper 24 bits of the data flow.

The "switch SWA" condition is present on the following:

Condition: XSF & (XSK) = 0,1,2,3,4,5,6,7

SWS rank B register (SWB)

The SWB register is a 60 bit clear/enter type register with gated clock pulse control. This is the second rank of the SWS. The SWB register is normally cleared and entered with data from the SWA register at the end of each clock period. This normal mode of operation is altered in the event of a SCM conflict, or in the case of executing a return jump instruction. The data in the SWB register is held in the event of a SCM conflict until the conflict is resolved. The conditions for clearing and entering new data in the SWB register are the same as those for the SWA register and are as follows:

Condition: NCF
or: NBF

"Switch SWB" condition

This condition is present only during the execution of a return jump instruction. It causes the data switch in the static network before the SWB register to switch from the SWA data path to an alternate path. This alternate path contains data from the RJX register which has been positioned in the 60 bit data word and supplemented with a constant background field as described under the return jump instruction in part 3 of this manual. The data switch is thrown for only one clock period. During this clock period the data in the RJX register is entered in the SWB register along with the background field. The "switch SWB" condition is defined as follows:

Condition: NBF & SXF

SWS rank C register (SWC)

The SWC register is the third rank of the SWS. It is a 60 bit clear/enter type register with gated clock pulse control. The SWC register is normally cleared and entered with data from the SWB register at the end of each clock period. This function is not performed when a SCM conflict exists. In this case the data in the SWC register is held until the NCF is reset. This condition is somewhat different from the condition for clearing and entering data in the SWA and SWB registers.

There is no data switch in the static network before the SWC register. This static network provides a delaying function to cover the short path problem described on page 2-1. The register is cleared and entered with new data on:

Condition: NCF

SWS rank D register (SWD)

The SWD register is the fourth rank of the SWS. It is a 60 bit clear/enter type register which is cleared at the end of every clock period. There is no clock pulse control on this register. The static network before the SWD register has no data switch and provides only a delaying function to cover the short path problem. The data in the SWC register is copied into the SWD register at the end of every clock period. This register begins that portion of the SWS which delays the data by an amount of time corresponding to the read access portion of the SCM bank read/write cycle.

SWS rank E register (SWE)

The SWE register is the fifth rank of the SWS. It is a 60 bit clear/enter type register which is cleared at the end of every clock period. There is no clock pulse control on this register. The static network before the SWE register contains a data switch which can select between the SWD register data and a data path from the I/O section of the CPU. This switch is controlled by a static condition "switch SWE" which originates in the SCM destination control unit. The switch normally selects the data from the SWD register. It is thrown to the data path from the I/O section for a single clock period when data from the I/O section is written into SCM. Data is entered in the SWE register at the end of every clock period from whichever source is selected by the data switch.

SWS rank F register (SWF)

The SWF register is the sixth rank of the SWS. It is a 60 bit clear/enter type register which is cleared at the end of every clock period. There is no clock pulse control on this register. The static network before the SWF register contains a data switch which can select between the SWE register data and a data path from the LCM section of the CPU. This switch is controlled by a static condition "switch SWF" which originates in the SCM destination control unit. The switch normally selects the data from the SWE register. It is thrown to the data path from LCM each clock period in which a LCM word is transmitted in a block copy mode. Data is entered in the SWF register at the end of every clock period from whichever source is selected by the data switch.

SWS rank G register (SWG)

The SWG register is the seventh and last rank of the SWS. It is a 65 bit clear/enter type register which is cleared at the end of every clock period. There is no clock pulse control on this register. The static network before the SWG register forms an odd parity bit for every 12 data bits in the 60 bit data word. The lowest order 12 data bits are summed as a group to form the lowest order parity bit. The next 12 data bits are summed to form the second parity bit. There are five parity bits formed in this manner. A parity bit is chosen zero if an odd number of bits in the associated 12 bit group have a value of one. A parity bit is chosen one if an even number of bits in the associated 12 bit group have a value of one.

There is no data switch in the static network before the SWG register. The 60 bits of data in the SWF register are transmitted to the SWG register at the end of every clock period along with the five newly formed parity bits. In the following clock period this data is either discarded or delivered to a bank write operand register in SCM.

SCM Data Distribution

Communication of data within the SCM system and with other parts of the CPU generally involves 65 bit parallel data transmission paths. These paths involve rather substantial static networks for merging of data and distribution of data. The logical interconnection of the data lines is illustrated in figure 2-23 on the following page.

There are eight bank read operand registers in the SCM system as described earlier in this section. Each of these registers serves four SCM banks. Each register is 65 bits in length. The eight sets of 65 bit data paths from these registers merge in a static network as illustrated in figure 2-23. There is no control or selection of these data paths. Only one SCM bank may be initiated in any one clock period. As a result, only one SCM bank can read data into a bank read operand register in any one clock period. This static network provides a common point for SCM read data from all 32 banks. This one 65 bit data word is then distributed to a number of destinations.

Data is transmitted to the IWS and to the X registers directly from the static merge network. The data is gated into the registers by control conditions appropriate for each register. Only the 60 useful data bits are transmitted to the IWS and the X registers. All 65 bits are transmitted to the SRO register as illustrated in figure 2-23. This register then relays the data to the other destinations in the following clock period.

A 65 bit static distribution network handles all data going to SCM banks. This network receives data from the last rank of the SWS. It also receives data from the bank read operand registers via the static merge network. The static distribution network contains a data switch which selects one of these two possible sources of data. The selection is based on the "write SCM" condition generated in the SCM destination control unit. This condition is present when a SCM bank is in the proper portion of its read/write cycle and data is to be written into that bank from the SWS. The switch is then thrown to the SWS information and that data is transmitted to the bank write operand register. When the switch is not thrown, the data read from a SCM bank is routed back to the bank write operand register for the same bank. This is the normal path for restoring data in the second half of a SCM read reference. The data from the static distribution network is routed to all 32 bank write operand registers. The data is gated into the proper register by the local bank sequence control.

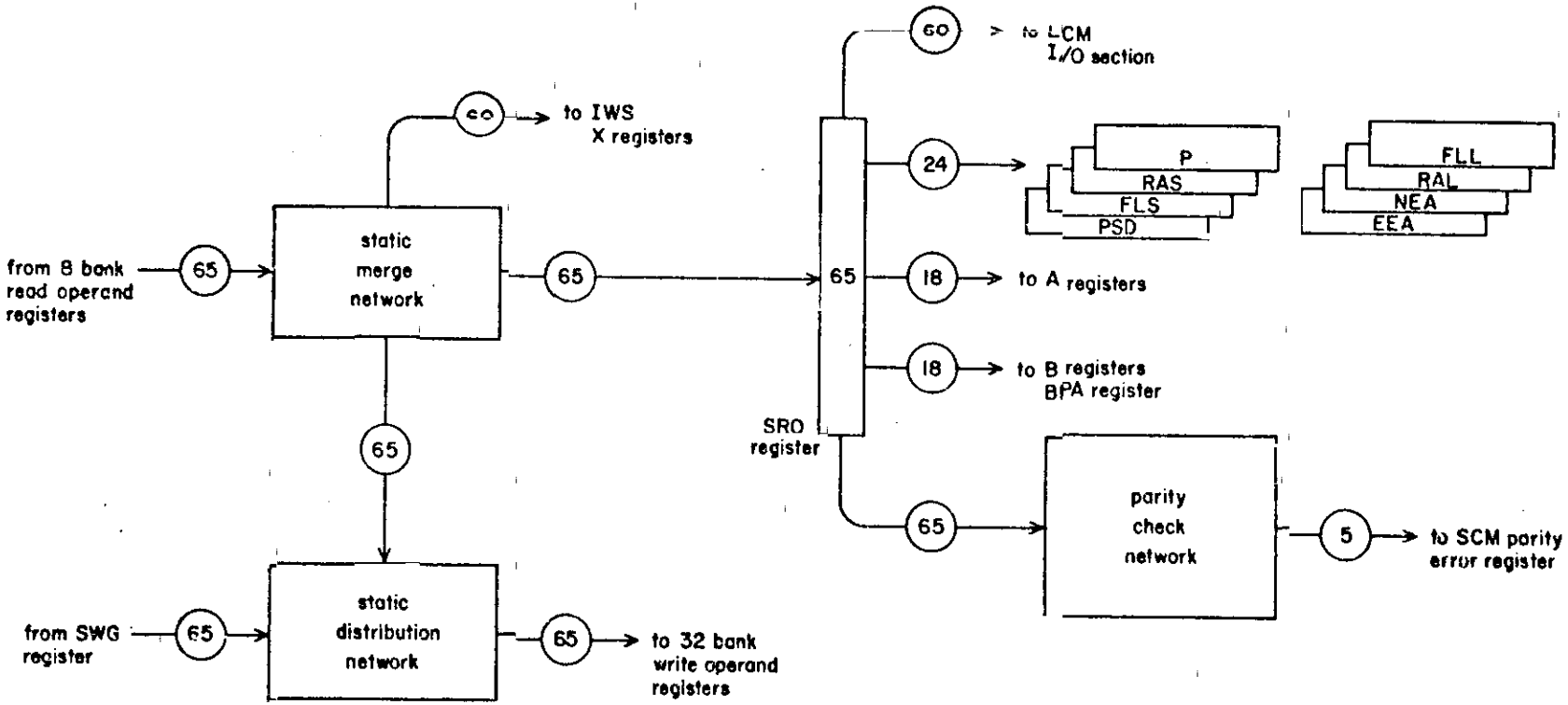


Fig. 2-23 SCM Data Distribution

Storage readout register (SRO)

The SRO register is a 65 bit clear/enter type register which is cleared at the end of every clock period. New data is entered in this register at the end of every clock period from the static merge network in the SCM data paths. This data corresponds to the data in a SCM bank read operand register if that bank has just completed the sample of the sense amplifier outputs. Only one SCM bank can be transmitting data on this network in a given clock period. If no SCM bank is transmitting data, the SRO register is entered with 65 bits of ones. This is the normal case when the SCM is idle. Data in the SRO register lags the data in the SCM bank read operand registers by one clock period.

The SRO register relays the SCM read data to other parts of the CPU. The 60 useful data bits are transmitted to the LCM and the I/O section of the CPU. The data is gated at the receiving end of these transmission paths by the appropriate sequence control. The highest order 24 bits of the 60 useful data bits are transmitted to the P register and to the supporting registers as illustrated in figure 2-23. The gating of this data to the proper register is controlled by the exchange sequence destination control unit. These paths are not used except during an exchange sequence. The lowest order 36 bits of the 60 useful data bits are transmitted to the A registers, B registers, and the BPA register. The gating of this data to the proper register is also under the control of the exchange sequence destination control unit.

The 65 bits in the SRO register are merged in a static parity check network. This network forms the modulo two sum of the 13 bits in each of five fields in the 65 bit word. Each field corresponds to the 13 bits in a SCM storage module. This is the reverse process of that formed in the rank G register of the SWS. The output of this static network is five bits of parity check information. A parity check bit has a zero value if the modulo two sum is odd. The parity check bit has a one value if the modulo two sum is even. These five bits are transmitted to the SCM parity error register. All five of these bits will have a zero value if the word read from SCM is correct. If any single bit in one of the 13 bit fields is incorrect, the corresponding parity check bit will have a one value. A clock period in which the SCM is idle will result in 65 one bits in the SRO register. This value is processed by the static parity check circuit in a normal manner and results in a parity correct indication.

SCM Destination Control Unit

The SCM destination control unit receives a four bit code from the SAS each clock period. This code is translated to determine the destination for each address processed by the SAS. The proper control condition is then generated to prepare the destination register for receiving the data from SCM. The address tags which are encoded into the four bits are listed on page 2-112. The translated control signals are illustrated in figure 2-24 on the following page.

The static translation network illustrated in figure 2-24 receives the address tag with the destination code directly from the SAS. This code is translated and one or more of the delay chains is set at the end of each clock period in which a valid address is transmitted from the SAS to a SCM bank address register. The delay chains consist of one bit registers separated by static delay networks to delay the control condition by the proper number of clock periods. Each box illustrated in figure 2-24 contains one bit of register which is cleared and entered with data from the preceding box at the end of every clock period. Each box then represents a one clock period delay for the associated control bit.

"Write SCM" condition

This condition is generated in the SCM destination control unit and controls the 65 bit data switch in the SCM data distribution network. When this condition exists the data in the SWS rank G register is routed to the SCM bank write operand registers. When this condition is absent the data in the SCM data merge network is routed to the SCM bank write operand registers. This condition is defined as follows:

Condition: Five clock period delay of destination code 02,07 & "go bank"
or: Five clock period delay of destination code 03,06,16,17 & "go bank" & no "range error"

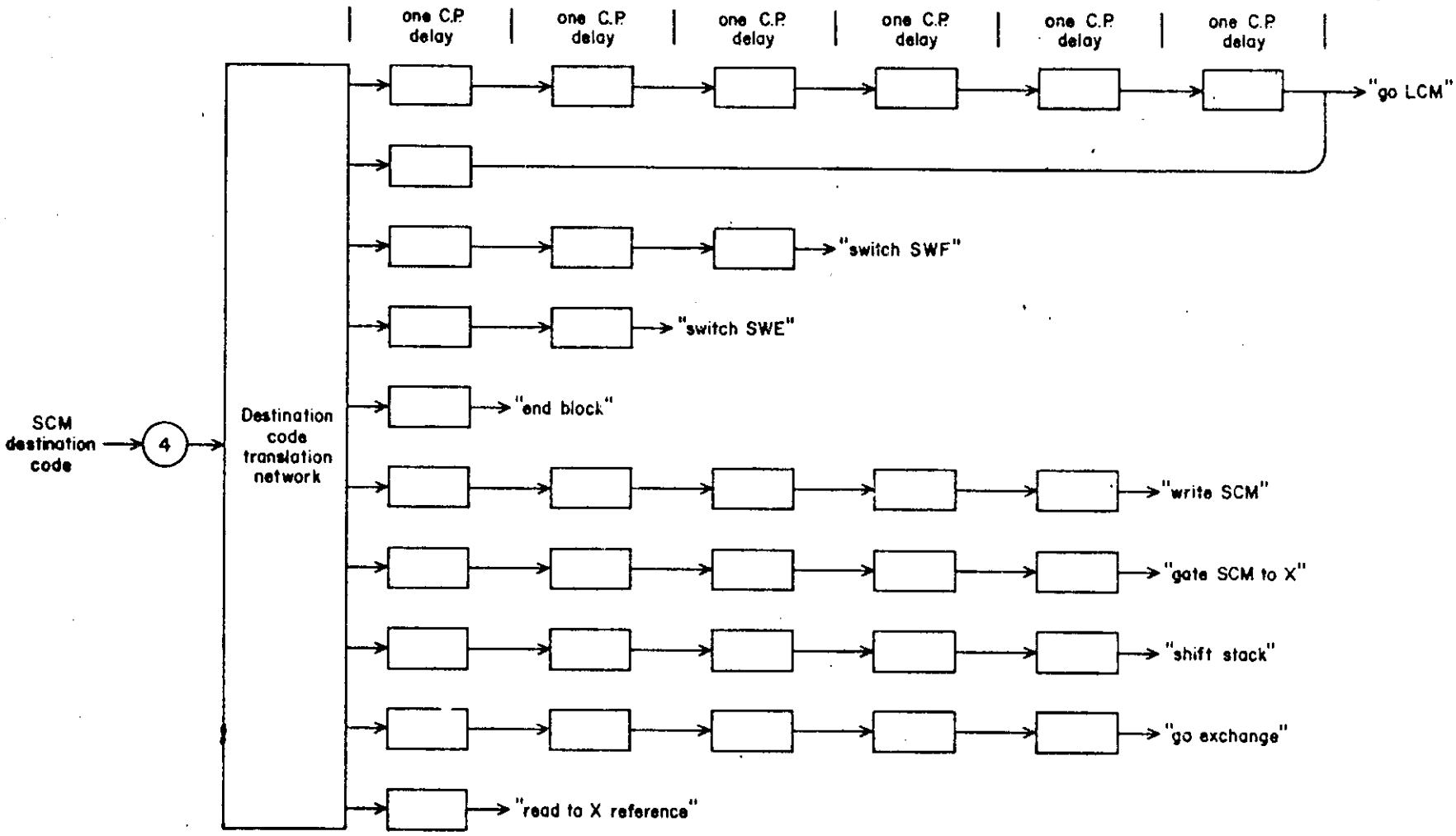


Fig. 2-24 SCM Destination Control

"Gate SCM to X" condition

This condition is generated in the SCM destination control unit and opens a gate for the 60 bit data path from the SCM data merge network to the X register data input path. The gate is open when the "gate SCM to X" condition is present. The gate is closed when the condition is absent. The condition is defined as follows:

Condition: Five clock period delay of destination code 02,11,12,13, 14,15 & "go bank"

"Shift stack" condition

This condition is generated in the SCM destination control unit and causes the IAS and IWS to shift data by one rank. The IAS receives a new address from the NSA register, and the IWS receives a 60 bit data word from the SCM data merge network. This condition is defined as follows:

Condition: Five clock period delay of destination code 01 & "go bank"

"Go exchange" condition

This condition is generated in the SCM destination control unit and is transmitted to the exchange sequence destination control unit. It causes the exchange sequence destination control unit to gate one word of an initiating exchange package to the proper destination registers and to advance the exchange package count register by one count. This condition is defined as follows:

Condition: Five clock period delay of destination code 02 & "go bank"

"Read to X reference" condition

This condition is generated in the SCM destination control unit and is transmitted to the X register access control unit. It causes the X register access control unit to enter the destination X register number in the access control chain. The condition is defined as follows:

Condition: One clock period delay of destination code 11,12,13,14, 15 & "go bank"

"Go LCM" condition

This condition is generated in the SCM destination control unit and is transmitted to the LCM control unit. It indicates to the LCM control unit that an address has gone to a SCM bank for a LCM block copy instruction. This condition is defined as follows:

Condition: Six clock period delay of destination code 04 & "go bank"
or: One clock period delay of destination code 06 & "go bank"

'Switch SWE" condition

This condition is generated in the SCM destination control unit and is transmitted to the SWS rank E register. The data switch before the SWS rank E register is thrown to the I/O section data path when this control condition is present. The data switch is thrown to the SWS rank D register data path when this control condition is absent. The condition is defined as follows:

Condition: Two clock period delay of destination code 07 & "go bank"

"Switch SWF" condition

This condition is generated in the SCM destination control unit and is transmitted to the SWS rank F register. The data switch before the SWS rank F register is thrown to the LCM data path when this control condition is present. The data switch is thrown to the SWS rank E register data path when this control condition is absent. The condition is defined as follows:

Condition: Three clock period delay of destination code 06 & "go bank"

"End block" condition

This condition is generated in the SCM destination control unit and is transmitted to the LCM control unit. It indicates to the LCM control unit that a block copy instruction has been executed to the point where all data references to SCM have cleared the SAS. The condition is defined as follows:

Condition: One clock period delay of destination code 00,01,02,03, 10,11,12,13

Exchange Destination Control Unit

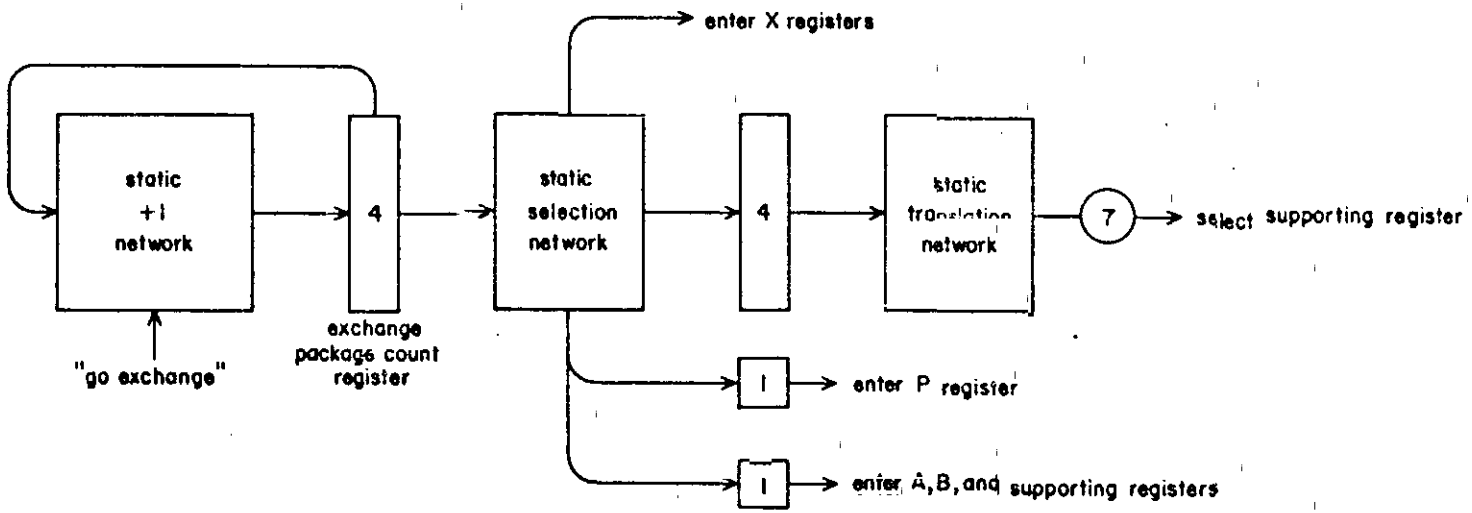
The exchange destination control unit is an appendage to the SCM destination control unit. It sequences the data in the initiating exchange package from SCM to the proper operating registers. Its only control input information comes from the SCM destination control unit in the form of the "go exchange" condition. This unit is illustrated in figure 2-25 on the following page.

The execution of an exchange sequence is controlled by two separate mechanisms. The terminating exchange package data is delivered to the SWS, and corresponding addresses to the SAS, under the control of the exchange sequence described on page 2-72. A four bit counter (XSK) in this control unit counts through the 16 steps of the terminating exchange package. The address codes for the 16 SCM references must pass through the SAS before reaching the exchange destination control unit to be described in this section. SCM bank conflicts may occur which will delay the addresses for the exchange sequence in the SAS. As a result the initiating exchange package data may not be delivered to the operating registers with a uniform lag in time from the terminating exchange package data. The exchange destination control must deliver data to the operating registers as specified by the timing of the SCM destination codes leaving the SAS.

The exchange destination control unit consists primarily of the 16 position counter illustrated in figure 2-25 and the translation networks to select the destination registers. The counter in this unit is cleared on a CPU "dead start" condition. The register content is advanced one count as each SCM destination code for an exchange reference leaves the SAS. The 16 such codes for an exchange sequence thus return the four bit register content to a zero value.

Exchange package count register

This is a four bit clear/enter type register with gated clock pulse control. It is cleared to zero during a CPU "dead start" condition. It is cleared and entered with a new count at the end of a clock period in which the "go exchange" condition is present. The new count is always one more than the previous value in the register modulo 16.



Exchange Destination Control

Exchange destination sequencing

The X register data in the initiating exchange package is delivered to the X registers under the control of the exchange package count register. The lowest order three bits of the exchange package count register specify the X register destination. The highest order bit in the exchange package count register is used as a control bit to request entry in the designated register. These four bits are transmitted to the X register access control unit. The X register data is transmitted directly from the SCM data merge network to the X register data input path. This data does not pass through the SRO register.

The remainder of the initiating exchange package data is processed through the SRO register and arrives at the destination register one clock period later in the sequence than the corresponding X register data. The control information for this data must also be delayed by one clock period. This function is performed by a second four bit register in the exchange destination control unit as illustrated in figure 2-25. The content of the exchange package count register is copied into this second four bit register at the end of every clock period. This second register is a clear/enter type register which is cleared at the end of every clock period. The lowest order three bits of this second register are transmitted to the A register access control unit and to the B register access control unit to specify register number.

A one bit flag in the exchange destination control unit is used for control of data entry in the P register. This is a clear/enter type bit register which is cleared at the end of every clock period. It is set for one clock period when the exchange package data for the P register is in the upper portion of the SRO register. The condition for setting this flag is the presence of the "go exchange" condition and a zero value in the exchange package count register.

A one bit flag in the exchange destination control unit is used for control of data entry in the A, B, and supporting registers. This flag is set at the end of a clock period when the "go exchange" condition is present and the highest order bit in the exchange package count register is zero. It is a clear/enter type bit register which is cleared at the end of every clock period. This flag is transmitted to the A register access control unit and to the B register access control unit to request entry in the selected register. This flag also controls the data entry in the supporting registers RAS, FLS, PSD, RAL, FLL, EEA, and NEA. A special static translation network in the exchange destination control unit provides the selection for these seven registers. The BPA register is treated as B register 0 in the processing of the initiating exchange package data.

Input/Output Section

The input/output (I/O) section of the CPU contains 15 full duplex channels for communication with the PPU and the mechanism for control and data transfer to SCM. This section of the CPU is described in the following portion of this manual. The description is divided into several sections by function. The overall organization of these sections is illustrated in figure 2-26 on the following page.

Each channel provides an input data path from a PPU. This is a 12 bit parallel transmission path with associated control lines as described in part 4 of this manual. The channel data is assembled into a 60 bit input register for transmission to the SCM buffer area for that channel. The 60 bit data transmission is time shared through an input data merge network common to all 15 channels. The SCM buffer address is maintained in the channel input control unit, and this unit interrupts the CPU program on a buffer threshold address or on an end of record signal from the PPU.

Each channel provides an output data path to a PPU. Data in a 60 bit disassembly register is shifted off into a 12 bit parallel data transmission path. The channel data register is refilled from the associated SCM buffer area as required to satisfy the PPU data requirements. The data is transmitted from SCM to the 60 bit channel disassembly register via the output data distribution network illustrated in figure 2-26. This distribution network is time shared by all 15 channels. The SCM buffer address for the output data is maintained in the channel output control unit. This unit interrupts the CPU program on a buffer threshold address.

The MCU communicates with SCM over a special set of data paths which are not illustrated in figure 2-26. This unit utilizes the input data merge network and output data distribution network in much the same manner as a normal channel. There is no buffer area in SCM for the MCU, however, and the control is handled in a special unit which is not like the normal channel control units. The MCU communication is mentioned in this section as if a channel 0 existed for this purpose. The details of the MCU to SCM control are treated in part 6 of this manual.

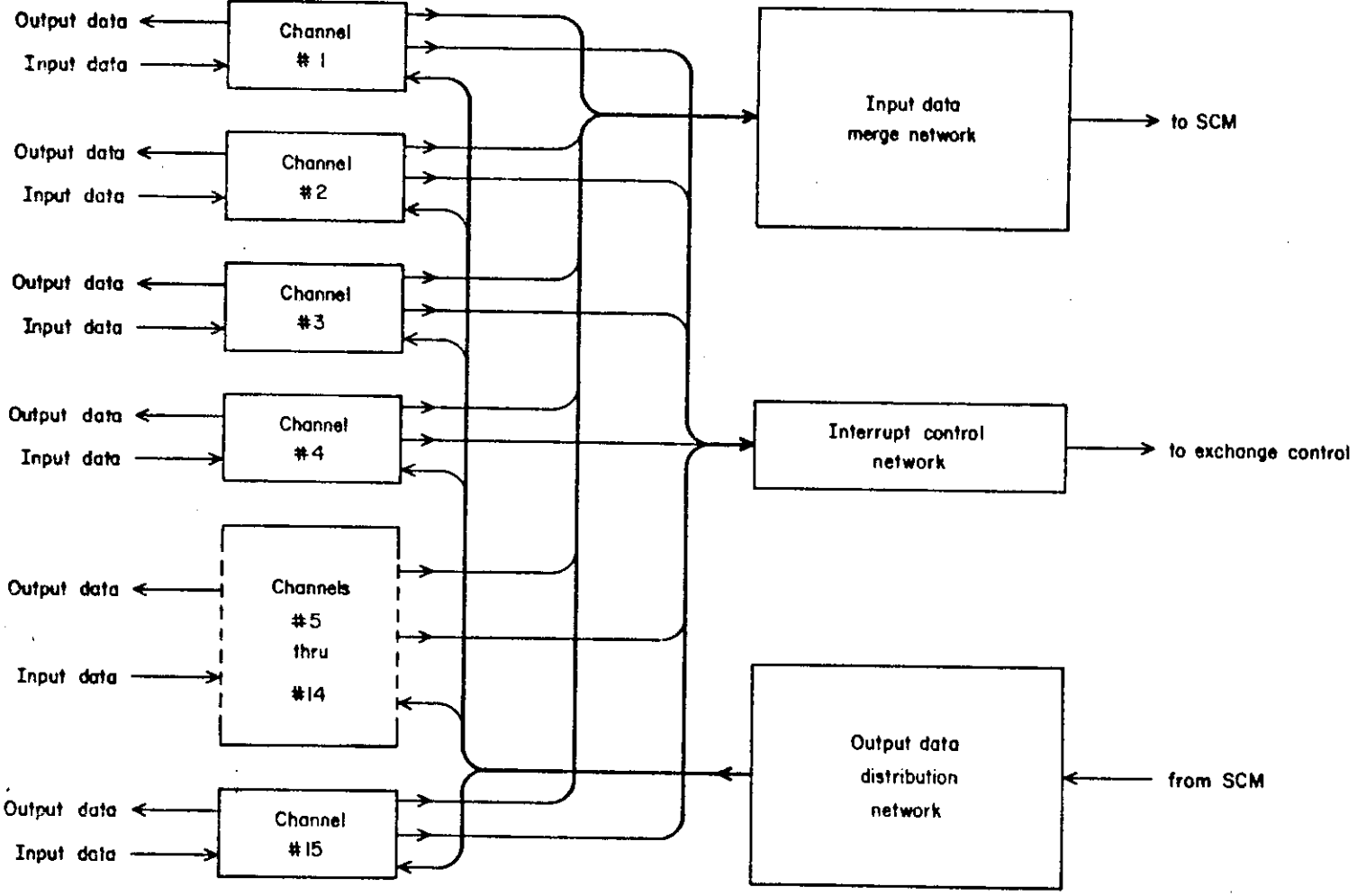


Fig. 2-26 I/O Section Organization

Channel input control unit

Each of the 15 input/output channels has a channel input control unit which is independent of the other channels. This control unit handles the control signals associated with the input data path from a PPU. It is not involved with the output data path which is handled in another control unit. The channel input control unit includes a parcel counter for the five parcels in a 60 bit word and an input buffer address counter for the current input address in the SCM buffer area. These two counters plus the associated control flags are illustrated in figure 2-27 on the following page.

Input address register

The input address register is a 12 bit clear/enter type register with gated clock pulse control. A portion of this register is connected to a static network which advances the content of the register by one count each time it is cleared and entered with new data. This portion of the register corresponds to the size of the buffer area in SCM reserved for the input buffer function. Figure 2-27 illustrates a seven bit counter for a 128 word SCM buffer area. This is the most common size buffer area. Other buffer sizes may be obtained by different hardware configurations, but all must be a power of two in size. The highest order bits in the input address register are wired to a constant value by pluggable wires in the CPU chassis. These bits determine the location of the SCM buffer area in the absolute address structure. All buffer areas must be located in the lowest 4096 addresses in SCM. Each buffer area must begin at an address which is a multiple of the power of two represented by the buffer size. The location of a particular buffer area in the SCM address structure may be changed by moving wires in the CPU chassis. The size of the buffer area may be changed by replacing the channel input control unit module with a module with a different input address register configuration.

The input address register is cleared to zero (in the counting portion) at the end of a clock period in which the input reset flag is set. This occurs whenever a CPU program in a monitor mode executes a reset input buffer instruction. The content of the input address register is advanced one count modulo buffer size at the end of a clock period in which the input word resume flag is set. This occurs whenever a 60 bit word has been delivered to SCM from the associated input data path.

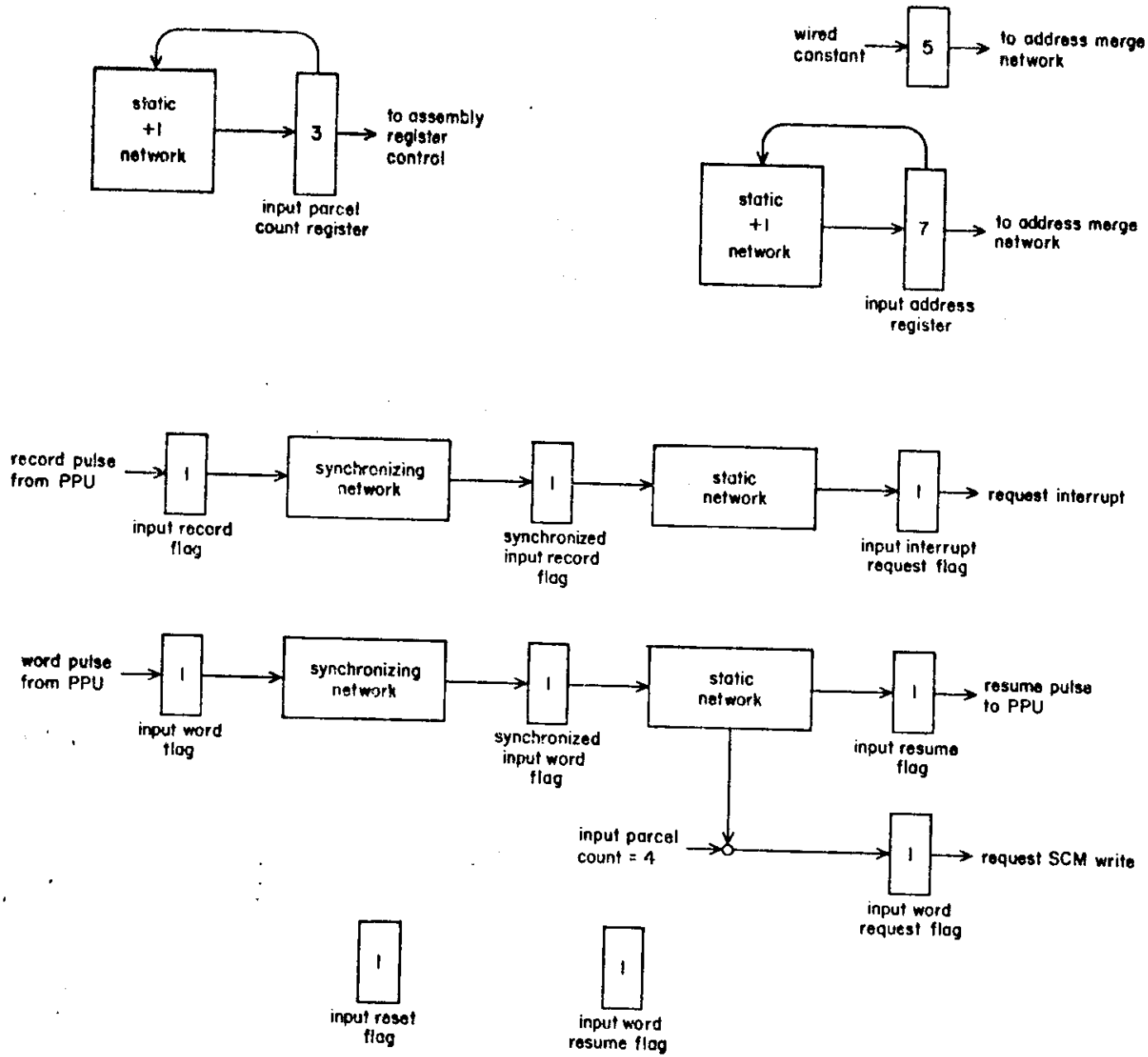


Fig. 2-27 Channel Input Control Unit

Input parcel count register

The input parcel count register is a three bit clear/enter type register which is cleared at the end of every clock period. New data is entered in this register at the end of every clock period from a static network which conditionally advances the register value by one count. This register holds the parcel count for the next 12 bit parcel in the 60 bit assembly register. The highest order parcel in the assembly register corresponds to a zero value in the input parcel count register. The lowest order parcel in the assembly register corresponds to a value of four in the input parcel count register. The input parcel count register is cleared to zero at the end of a clock period in which the following condition exists.

Condition: input word request flag
or: input reset flag

The input parcel count register is reset to a value one count larger than its previous value at the end of a clock period in which the following condition exists.

Condition: synchronized input word flag & no synchronized input record flag & no input word request flag & no interrupt lockout condition

The input parcel count register is reset to its previous value at the end of all clock periods other than those defined in the two conditions above.

Input reset flag

The input reset flag is a one bit clear/enter type register which is cleared at the end of every clock period. This flag is set at the end of a clock period in which an input reset condition exists and the channel number for this input control unit is selected. These conditions will be present for one clock period on execution of a CPU instruction 0160 (reset input buffer) with the proper channel number selected in register Bk.

Input word flag

The input word flag is a one bit register with a separate set and a separate clear input. This flag is set on the arrival of a word pulse over the associated input data path. This pulse is generated in a PPU when data is transmitted over the input data path. The pulse is not necessarily synchronous with the CPU clock since the PPU may be remote from the CPU main frame and operating from a separate clock source. This flag is cleared during the clock period in which the input resume flag is set. This flag is also cleared on an I/O clear condition for the CPU. This condition is programed in the MCU on a CPU dead start only.

Input record flag

The input record flag is a one bit register with a separate set and a separate clear input. This flag is set on the arrival of a record pulse over the associated input data path. This pulse is generated in a PPU when a 74 instruction is executed. The pulse is not necessarily synchronous with the CPU clock since the PPU may be remote from the CPU main frame and operating from a separate clock source. This flag is cleared at the end of a clock period in which the following condition exists:

Condition: synchronized input record flag & no input word request flag & no input interrupt request flag & input parcel count = 0

This flag is also cleared on an I/O clear condition for the CPU. This condition is programed in the MCU on a CPU dead start only.

Synchronized input word flag

This flag is a one bit clear/enter type register which is cleared at the end of every clock period. It is set at the end of a clock period in which the synchronize condition is present and the input word flag has been set for three clock periods. The synchronize condition occurs every fourth clock period. The synchronizing network between the input word flag and the synchronized input word flag introduces a three clock period delay. The setting of the synchronized input word flag therefore lags the setting of the input word flag by a minimum of three clock periods and a maximum of six clock periods depending on the time of arrival of the word pulse with respect to the synchronize condition.

Synchronized input record flag

This flag is a one bit clear/enter type register which is cleared at the end of every clock period. It is set at the end of a clock period in which the synchronize condition is present and the input record flag has been set for three clock periods. The synchronize condition occurs every fourth clock period. The synchronizing network between the input record flag and the synchronized input record flag introduces a three clock period delay. The setting of the synchronized input record flag therefore lags the setting of the input record flag by a minimum of three clock periods and a maximum of six clock periods depending on the time of arrival of the record pulse with respect to the synchronize condition.

Input interrupt request flag

This flag is a one bit register with a separate set and a separate clear input. It is set whenever the buffer associated with this input data path has been filled to an interrupt threshold address. There are two such addresses: one located in the center of the buffer area, and one located at the end of the buffer area. This flag is also set whenever a record pulse has been received from the PPU and has been synchronized. The precise condition for setting the input interrupt request flag for a 128 word buffer is at the end of a clock period in which the following condition is satisfied.

Condition: lowest order six bits in the input address register all have a value of one & an input word resume condition exists for this channel

or: synchronized input record flag & no input word request flag & no input interrupt request flag & input parcel count = 0

The input interrupt request flag is cleared at the end of a clock period in which an input interrupt resume condition exists for this channel. This flag is also cleared on an I/O clear condition for the CPU. This condition is programed in the MCU on a CPU dead start only.

Input resume flag

The input resume flag is a one bit clear/enter type register which is cleared at the end of every clock period. This flag is set for one clock period, and generates a one clock period wide resume pulse, when the incoming data from the PPU has been accepted in the 60 bit input assembly register. The resume pulse formed by this flag is transmitted to the PPU over the associated input data path. This flag is normally set a few clock periods after the receipt of the word pulse from the PPU. The minimum time is four clock periods. The maximum time may be very long, however, because of backup conditions which may exist in SCM. This flag is set at the end of a clock period in which the following condition exists.

Condition: synchronized input word flag & no synchronized input record flag & no input word request flag & no interrupt lockout condition

Input word request flag

The input word request flag is a one bit register with a separate set and a separate clear input. It is set whenever the 60 bit assembly register associated with this channel input control unit has been filled and is ready for entry in the SCM buffer area. This flag is set at the end of a clock period in which the following condition exists.

Condition: synchronized input word flag & no synchronized input record flag & no input word request flag & no interrupt lockout condition & input parcel count = 4

or: synchronized input record flag & no input word request flag & no input interrupt request flag & input parcel count is not zero

The input word request flag is cleared at the end of a clock period in which an input word resume condition is present for this channel. This condition occurs when SCM has received the 60 bit entry from the input assembly register. The input word request flag is also statically cleared by the I/O clear condition for the CPU. This condition is programed in the MCU on a CPU dead start only.

Input word resume flag

This flag is a one bit clear/enter type register which is cleared at the end of every clock period. This flag is set at the end of a clock period in which an input word resume condition is present for this channel. This condition occurs when the SCM has received a 60 bit data word from the associated input assembly register. The input word resume flag remains set for only one clock period.

Normal input sequence

The following description lists chronologically the events in a normal record input sequence. The sequence begins with a CPU instruction which resets the channel input control unit for receipt of a new record. This is an 0160 instruction with the channel number in B register k. The input reset condition sets the input reset flag in the channel input control unit. At the end of the following clock period the input parcel count register and the input address register are cleared. The channel input control unit is now ready for a new record beginning with the first word in the SCM buffer area.

The PPU connected to the input channel must next be notified that the input buffer is ready to receive data. This must be accomplished by the transmission of a message over the associated output channel. The PPU then enters the first 12 bit word in its output register. This entry causes the transmission of a word pulse to the input word flag in the channel input control unit. Between three and six clock periods later the synchronized input word flag will set. At the end of the following clock period the 12 bits of data are sampled into the upper parcel of the 60 bit assembly register and the input resume flag is set. At this same time the content of the input parcel count register is advanced from zero to one.

The input resume flag sends a one clock period wide resume pulse to the PPU. This resume pulse is synchronized at the PPU and clears the word flag in the PPU. The second 12 bit word may now be entered in the PPU output register. The sequence of word pulse and resume pulse continues as each 12 bit word is transmitted over the data path. The next significant event occurs as the fifth word pulse arrives at the channel input control unit and is synchronized. At this time the input parcel count register contains a value of four. The input data is sampled into the lowest order parcel of the 60 bit assembly register. In this case the input resume flag is set and the input word request flag is also set. A resume pulse is transmitted to the PPU and an input word request condition exists in the CPU.

The input word request flag clears the input parcel count register in preparation for the arrival of the next PPU word. However, the input word request flag blocks the processing of a new 12 bit word should one arrive while the input word request flag is set. The input word request flag remains set until the 60 bit word in the input assembly register has been accepted by SCM. This may be only a few clock periods, or it may be many clock periods, depending on the SCM bank conflicts and higher priority SCM references that may exist. When the 60 bit word has been accepted by SCM the input word resume flag is set in the channel input control unit. At the same time the input word request flag is cleared. In the following clock period the content of the input address register is advanced one count to the second address in the SCM buffer area and the 60 bit assembly register is cleared. The first 60 bit word has now been stored in the first location in the SCM buffer area.

As soon as the input word request flag clears, the next 12 bit PPU word may be entered in the highest order parcel of the 60 bit assembly register. The sequence of word pulse and resume pulse communication between the PPU and the channel input control unit continues until the 60 bit assembly register has again been filled. The process of setting the input word request flag and transmitting the 60 bit word to SCM then repeats. The next significant event occurs as a 60 bit word is stored in SCM with the input address register set to a value so that the lowest order six bits are all one. At this time the input interrupt request flag is set.

The input interrupt request flag does not normally interfere with the operation of the rest of the channel input control unit. The communication between the PPU and the channel input control unit continues; 12 bit words are assembled into 60 bit words, and the 60 bit words are stored in SCM. The input interrupt request flag causes an I/O interrupt request condition in the CPU which is independent of the I/O word requesting mechanism. The I/O interrupt request condition causes a CPU exchange sequence to an exchange package address associated with the input channel. A CPU program resulting from the exchange sequence then processes the data in the first half of the SCM buffer area for the input channel. The input interrupt request flag remains set until the CPU program has been completed and an exchange exit instruction returns the CPU to its previous program.

Terminating an input record

An input record may contain data of any length. The buffer in SCM operates in a circular mode with threshold interrupt addresses in the center of the buffer area and at the end of the buffer area. The record is terminated by the transmitting PPU at any 12 bit word. The PPU program terminates the record by setting the record flag for the PPU output channel. The PPU program must sense the output channel word flag to make sure that the word flag has been cleared by a resume pulse before setting the record flag.

The PPU transmits a record pulse over the input data path to the CPU when the record flag is set in the PPU. This pulse sets the input record flag in the channel input control unit. Several clock periods later the synchronized input record flag is set. This begins the process of terminating the input record by completing the data in the SCM buffer area.

The input assembly register may contain one or more parcels of data at the time the input record flag is set. This constitutes a partial 60 bit word which must be stored in the SCM buffer area before interrupting the CPU program. The partial word will have the data left justified with zero parcels in the lowest order bit positions. The input word request flag is set by the combination of a synchronized input record flag and an input parcel count not equal to zero. If the parcel count is zero at the time of the input record flag, no partial word exists. The input word request flag causes the entry of the partial 60 bit word in the SCM buffer in a normal manner. When this entry has been made, and the input word request flag has been cleared, the input interrupt request flag sets to initiate the interrupt of the CPU program. The CPU program is able to distinguish the interrupt due to a record flag from that due to a SCM buffer threshold. It does this by sampling the SCM buffer address to determine if a threshold has been crossed since the last interrupt. The CPU program then recognizes the end of record situation and processes the remaining buffer data accordingly.

The PPU must not begin transmitting a new record of input data before the CPU has completed processing the data in the SCM buffer area. There is no hardware mechanism to prevent the PPU from beginning a new record. The PPU program must wait for a positive response from the CPU program over the associated output data path before beginning a new record. Should the PPU proceed before the CPU has reset the input buffer, the data would appear in the buffer as a continuation of the last record and would be partially lost.

Interrupt lockout condition

This is an abnormal condition which may occur in the channel input control unit when an interrupt request is not processed by the CPU program for a long time. This condition occurs when the input interrupt request flag is still set from the last buffer threshold when the next threshold is reached. This condition may be precisely stated as follows:

Condition: input interrupt request flag & lowest order six bits in the input address register all have a value of one

This condition prevents further processing of input data from the PPU until the CPU has cleared the work backlog. When the CPU program has completed processing the data in the SCM buffer associated with the first threshold interrupt, the input interrupt request flag will clear. It will reset in the following clock period for the data in the other half of the SCM buffer. The input data path is then released to resume input in a normal manner as the CPU processes the second interrupt request.

Short input records

An input record may be less than one 60 bit word in length. In this case the data is left justified in the 60 bit word and the lowest order parcels are filled with zeros. One word is stored in the SCM buffer in the same manner as for a longer record with a record length which is not a multiple of five 12 bit words.

An input record may be processed with a zero length. In this case the SCM buffer receives no data at all. The CPU program is able to detect the zero length record by the fact that an interrupt occurred and no data was present.

Input Data Merge Network

The input data merge network illustrated in the general I/O section diagram of figure 2-26 receives data from each of the 15 channel input assembly registers plus an assembly register for the MCU. This data merging network is illustrated in greater detail in figure 2-28 on the following page.

Each of the 15 input assembly registers associated with the I/O channels is controlled by the corresponding channel input control unit. This control unit sequences the entry of the five parcels of data into the 60 bit register and requests access to the SCM with the setting of the input word request flag. The SCM access control unit processes this request and enters an address in the SAS from the channel input address register. This address later emerges, and the associated address tag is interpreted by the SCM destination control unit. This last unit then selects the data path in the input data merge network and throws a switch in the SWS at the proper time to gate the 60 bits of merged input data into the rank E register of the SWS.

Input assembly register

The input assembly register is a 60 bit clear/enter type register with gated clock pulse control. There is one such register associated with each of the 15 I/O channels plus one associated with the MCU. The clock pulse control for these registers is divided into five sections of 12 bits each. In effect there are five registers, each of 12 bit length, in the input assembly register. Each of the five sections has its own clock pulse control for clearing and entering data in that portion of the register. The control information for this register originates in the channel input control unit. The input parcel count register designates which of the five sections will next be entered. The designated section is cleared and entered with new data at the end of a clock period in which the following condition exists.

Condition: synchronized input word flag & no synchronized input record flag & no input word request flag & no interrupt lockout condition

In addition to the above control, the lowest order four parcels of the input assembly register are statically cleared during the clock period in which the input word resume flag is set.

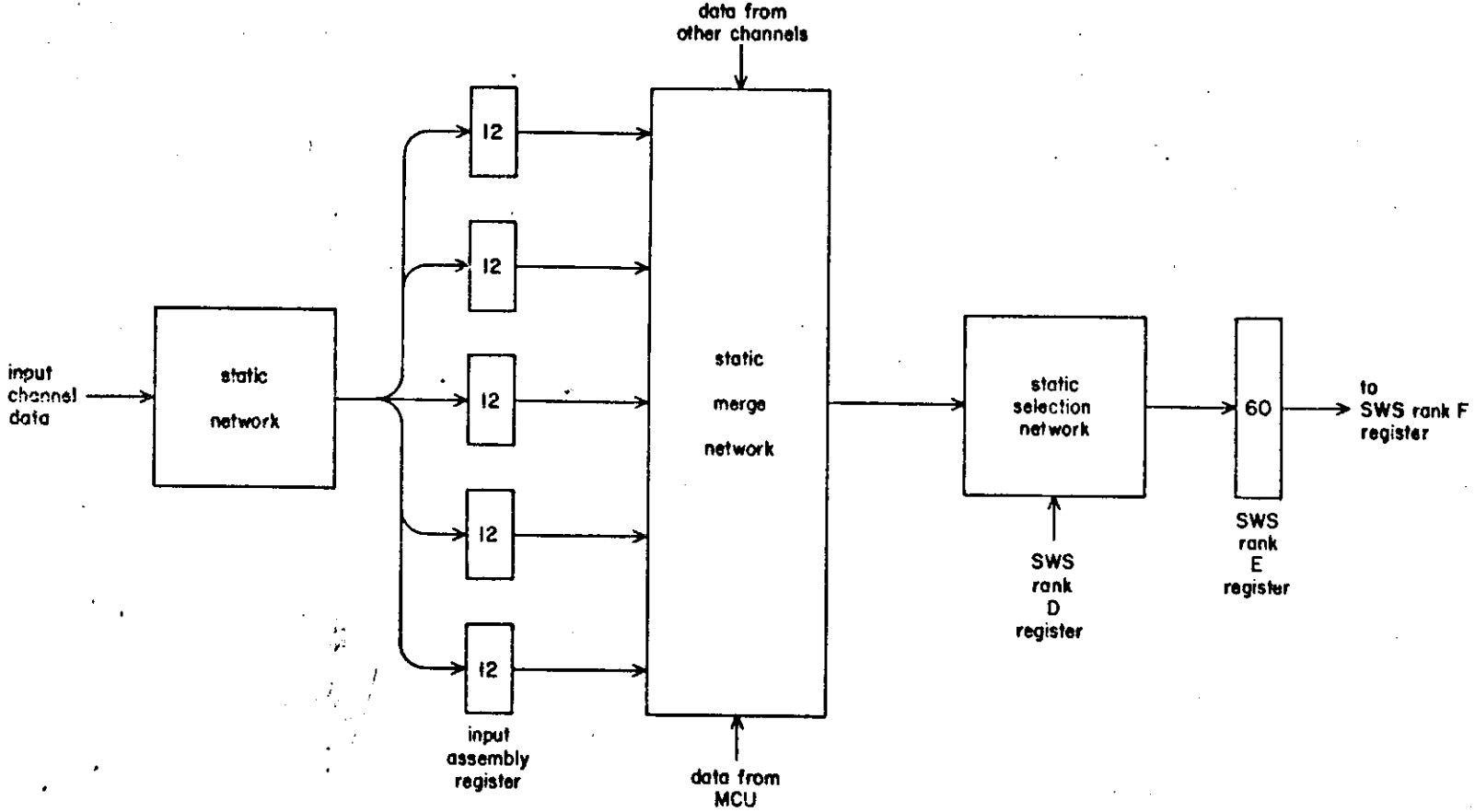


Fig. 2-28 Input Data Merge Network

Channel output control unit

Each of the 15 input/output channels has a channel output control unit which is independent of the other channels. This control unit handles the control signals associated with the output data path to a PPU. It is not involved with the input data path which is handled in another control unit. The channel output control unit includes a parcel counter for the five parcels in a 60 bit word and an output buffer address counter for the current output address in the SCM buffer area. These two counters plus the associated control flags are illustrated in figure 2-29 on the following page.

Output address register

The output address register is a 12 bit clear/enter type register with gated clock pulse control. A portion of this register is connected to a static network which advances the content of the register by one count each time it is cleared and entered with new data. This portion of the register corresponds to the size of the buffer area in SCM reserved for the output buffer function. Figure 2-29 illustrates a seven bit counter for a 128 word SCM buffer area. This is the most common size buffer. Other buffer sizes may be obtained by different hardware configurations, but all must be a power of two in size. The highest order bits in the output address register are wired to a constant value by plugable wires in the CPU chassis. These bits determine the location of the SCM buffer area in the absolute address structure. All buffer areas must be located in the lowest 4096 addresses in SCM. Each buffer area must begin at an address which is a multiple of the power of two represented by the buffer size. The location of a particular buffer area in the SCM address structure may be changed by moving wires in the CPU chassis. The size of the buffer area may be changed by replacing the channel output control unit module with a module with a different output address configuration.

The output address register is cleared to zero (in the counting portion) at the end of a clock period in which the output reset flag is set. This occurs whenever a CPU program in a monitor mode executes a reset output buffer instruction for this channel. The content of the output address register is advanced one count (modulo buffer size) at the end of a clock period in which the following condition exists.

Condition: synchronized output resume flag & no interrupt lockout condition & parcel count is zero

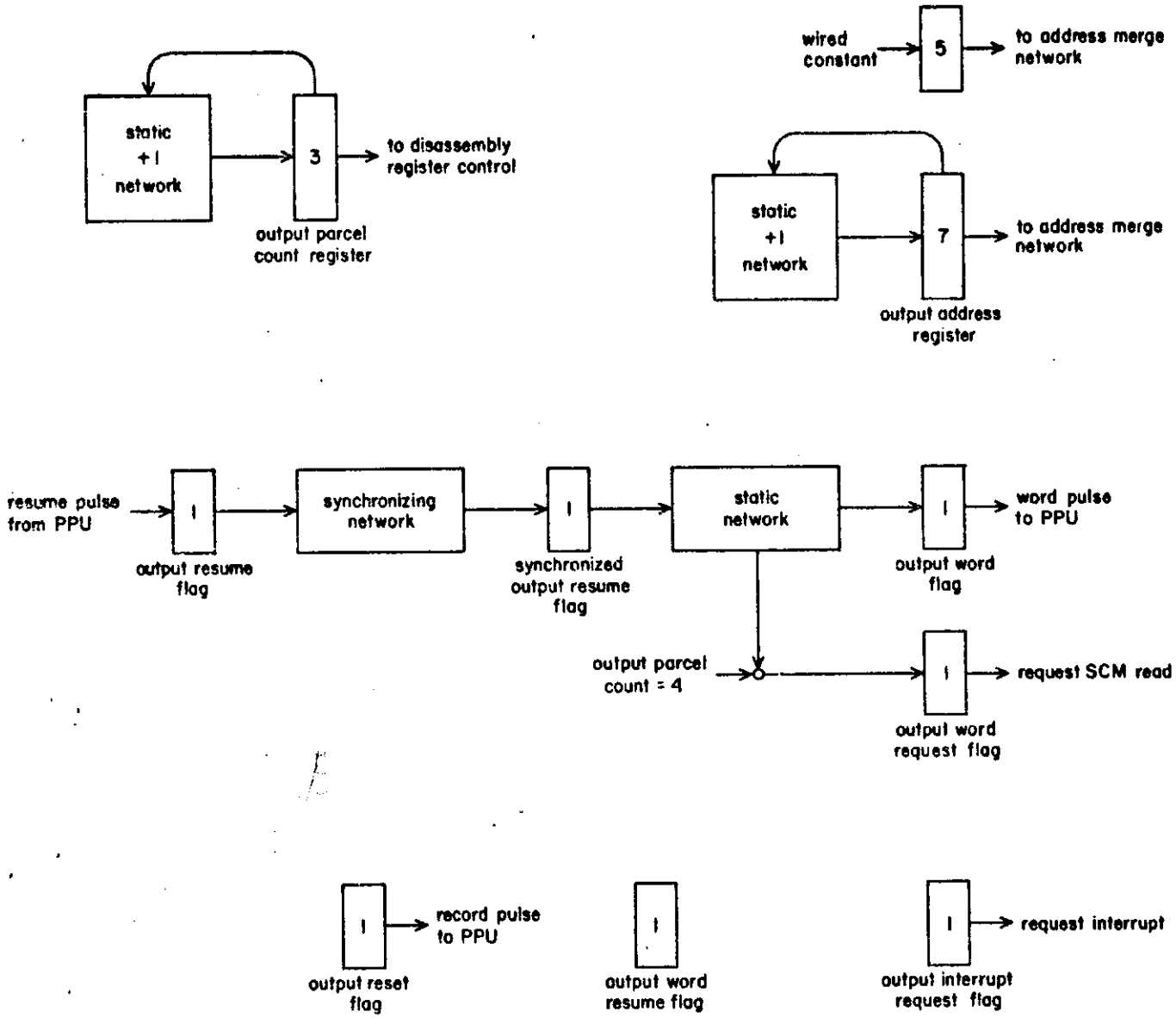


Fig. 2-29 Channel Output Control Unit

Output parcel count register

The output parcel count register is a three bit clear/enter type register which is cleared at the end of every clock period. New data is entered in this register at the end of every clock period from a static network which conditionally advances the register content by one count. This register holds the parcel count for the next 12 bit parcel in the 60 bit disassembly register. The highest order parcel in the disassembly register corresponds to a zero value in the output parcel count register. The lowest order parcel in the disassembly register corresponds to a value of four in the output parcel count register. The output parcel count register is statically cleared during those clock periods in which the output word request flag is set. It is reset to a value one count larger than its previous value at the end of a clock period in which the following condition exists.

Condition: synchronized output resume flag & no interrupt lockout condition

The output parcel count register is reset to its previous value at the end of all clock periods other than those defined above.

Output reset flag

The output reset flag is a one bit clear/enter type register which is cleared at the end of every clock period. This flag is set at the end of a clock period in which an output reset condition exists for this channel. This condition will be present for one clock period on execution of an 0170 instruction with the proper channel number selected in register Bk.

The output reset flag will remain set for one clock period. During that clock period the channel output control unit will transmit a one clock period wide record pulse to the PPU over the associated output data path. This pulse will set the record flag for the PPU input channel to indicate that a record has been completed and a new record will begin.

Output resume flag

The output resume flag is a one bit register with a separate set and a separate clear input. This flag is set on the arrival of a resume pulse from the PPU connected to the output data path. This pulse is generated in the PPU when the data on the channel is sampled into the PPU registers by an input instruction. The pulse is not necessarily synchronous with the CPU clock since the PPU may be remote from the CPU main frame and operating from a separate clock source. This flag is cleared at the end of a clock period in which the following condition exists.

Condition: synchronized output resume flag & no interrupt lockout condition

This flag is also cleared statically by an I/O clear condition for the CPU. This condition is programed in the MCU on a CPU dead start only.

Synchronized output resume flag

This flag is a one bit clear/enter type register which is cleared at the end of every clock period. It is set at the end of a clock period in which the synchronize condition is present and the output resume flag has been set for three clock periods. The synchronize condition is generated by the CPU clock period counter and occurs every fourth clock period. The synchronizing network between the output resume flag and the synchronized output resume flag introduces a three clock period delay. The setting of the synchronized output resume flag therefore lags the setting of the output resume flag by a minimum of three clock periods and a maximum of six clock periods, depending on the time of arrival of the resume pulse with respect to the synchronize condition.

Output word flag

The output word flag is a one bit clear/enter type register which is cleared at the end of every clock period. This flag is set for one clock period, and generates a one clock period wide word pulse, when a 12 bit parcel of data in the disassembly register is ready for transmission to the PPU. The word pulse generated by this flag is transmitted to the PPU where it sets the word flag for the PPU input channel. The associated data from the disassembly register may then be sampled into the PPU registers.

The output word flag is normally set a few clock periods after the receipt of a resume pulse from the PPU. The minimum time is four clock periods. The maximum time is very long, however, because of possible backup conditions in SCM which may delay the reading of a new 60 bit word to the disassembly register. This flag is set at the end of a clock period in which the following condition exists.

Condition: output word resume flag
or: synchronized output resume flag & no interrupt lockout condition & parcel count not equal four

Output word request flag

The output word request flag is a one bit register with a separate set and a separate clear input. It is set whenever the data in the 60 bit disassembly register has been transmitted to the PPU and a new word is required from SCM. This flag is set at the end of a clock period in which the following condition exists.

Condition: output reset flag
or: synchronized output resume flag & no interrupt lockout condition & parcel count equal four

The output word request flag is cleared at the end of a clock period in which the output word resume condition is present for this channel. This condition occurs when SCM has delivered a new 60 bit word to the disassembly register. The output word request flag is also statically cleared by the I/O clear condition for the CPU. This condition is programed in the MCU on a CPU dead start only.

Output word resume flag

This flag is a one bit clear/enter type register which is cleared at the end of every clock period. This flag is set at the end of a clock period in which the output word resume condition is present for this channel. This condition occurs when SCM has delivered a 60 bit word to the SRO register which is intended for transmission to the output channel disassembly register. The output word resume flag remains set for only one clock period. During this clock period the data is transmitted to the 60 bit disassembly register.

Output interrupt request flag

This flag is a one bit register with a separate set and a separate clear input. It is set whenever the buffer associated with this output data path has been emptied to an interrupt threshold. There are two such addresses: one located in the center of the buffer area, and one located at the end of the buffer area. The precise condition for setting the output interrupt request flag for a 128 word buffer is at the end of a clock period in which the following condition is satisfied.

Condition: lowest order six bits in the output address register all have a value of one & parcel count is zero & synchronized output resume flag & no interrupt lockout condition

This flag is cleared at the end of a clock period in which an interrupt resume condition exists for this channel. The interrupt resume condition will be present for one clock period when the CPU program associated with the interrupt request for this channel has been completed and is terminating with an exchange exit instruction. This flag is also cleared on an I/O clear condition for the CPU. This condition is programed in the MCU on a CPU dead start only.

Interrupt lockout condition

This is an abnormal condition which may occur in the channel output control unit when an interrupt request is not processed by the CPU program for a long time. This condition occurs when the output interrupt request flag is still set from the last buffer threshold when the next threshold is reached. This condition may be precisely stated as follows:

Condition: output interrupt request flag & lowest order six bits in the output address register all have a value of one & parcel count is zero

This condition prevents further processing of output data to the PPU until the CPU has cleared the work backlog. When the CPU program has completed processing the first threshold interrupt request, the output interrupt request flag will clear. It will reset again for the new interrupt request. The output data path is then released for the data in the SCM buffer associated with the first interrupt request.

Normal output sequence

The following description lists chronologically the events in a normal output record sequence. The sequence begins with a CPU instruction which resets the channel output control unit. This is an 0170 instruction with the channel number in register Bk. The output reset condition resulting from execution of this instruction is present for one clock period. At the end of this clock period the output reset flag is set in the channel output control unit.

The output reset flag remains set for only one clock period. At the end of this clock period the counting portion of the output address register is cleared to zero. This resets the buffer address to the first location in the SCM buffer. At this same time the output word request flag is set. This flag then requests a SCM reference to read the first word from the SCM buffer to the output disassembly register.

The output parcel count register is statically cleared to zero by the output word request flag. Some number of clock periods later, when SCM is ready to deliver the 60 bit word to the output disassembly register, an output word resume condition will be present for this channel. This condition lasts for one clock period, and the output word request flag is cleared at the end of this clock period. The output word resume flag is set at this same time.

The output word resume flag causes two actions in the channel output control unit. At the end of the one clock period in which the output word resume flag is set, the 60 bit disassembly register is cleared and entered with data from SCM. At this same time the output word flag is set. The output word flag generates a one clock period wide word pulse which is transmitted to the PPU over the output data channel. Concurrent with the word pulse, the upper parcel of the data in the disassembly register is transmitted to the PPU. This data remains static on the 12 data lines as long as the parcel counter has a zero value. The word pulse sets the PPU input channel word flag when it arrives at the PPU. This flag then allows the PPU to sample the 12 data lines and transmit a resume pulse to the channel output control unit.

The output resume flag in the channel output control unit is set when the resume pulse from the PPU arrives at the CPU. This flag is then synchronized and the synchronized output resume flag is set. This last flag then sets in motion the mechanism for advancing the disassembly register to the next parcel position.

The synchronized output resume flag is set for one clock period. At the end of this clock period a number of actions occur. The output parcel count register is entered with a new parcel count one greater than the previous value. In this case the parcel count will advance from zero to one. This change in parcel count switches the static network associated with the 60 bit disassembly register so that the second parcel of the register content is transmitted to the PPU over the output data path. At this same time the content of the output address register is advanced to the next address in the SCM buffer (modulo buffer size). The advance of the buffer address occurs at this time rather than earlier in order that the CPU program in monitoring the channel status may determine when the PPU has accepted the first parcel of a new 60 bit word. The output resume flag is cleared and the output word flag is set at the end of this clock period.

The output word flag transmits a one clock period wide word pulse to the PPU for the second parcel of the data in the disassembly register. The cycle of communication then repeats. The parcel count is advanced each time the synchronized output resume flag is set. The output address register content is altered only for a parcel count of zero. The next special action occurs when the synchronized output resume flag sets and the parcel count is four. At this time the parcel count is advanced to five and the output resume flag is cleared. The output word flag is not set, however, and the output word request flag is set instead.

The output word request flag clears the parcel count to zero and requests a SCM reference for the next word in the SCM buffer area. Some number of clock periods later the output word resume condition appears and the cycle of delivering a new 60 bit word to the PPU begins. This process continues until the content of the output address register has been advanced to the point where the lowest order six bits all have a value of one. When the synchronized output resume flag is set, the parcel count is zero, and the lowest order six bits of the address are ones, the output interrupt request flag is set in addition to the previously described actions.

The output interrupt request flag does not normally interfere with the operation of the rest of the channel output control unit. The communication between the channel output control unit and the PPU continues; 60 bit words are disassembled into 12 bit words, and the 12 bit words are delivered to the PPU. The output interrupt request flag causes an I/O interrupt request condition in the CPU which is independent of the I/O word requesting mechanism. The I/O interrupt request condition causes a CPU exchange sequence to an exchange

package address associated with the output channel. A CPU program resulting from the exchange sequence then refills the portion of the SCM buffer area which has been emptied by the channel output control unit. The output interrupt request flag remains set until the CPU program has been completed and an exchange exit instruction returns the CPU to its previous program.

Terminating an output record

The length of an output record is determined by the receiving PPU and not by the transmitting CPU program. The PPU program must know the record length by prearranged convention or by data content of the transmitted record. When the PPU has received the expected amount of data it simply stops reading data from the PPU input channel. This stops further transmitting action on the part of the channel output control unit. The PPU program must sense the record flag on the PPU input channel to determine when the CPU program has cleared the SCM buffer area and begun a new record transmission.

Output Data Distribution Network

The output data distribution network illustrated in the general I/O section diagram of figure 2-26 delivers data to the 15 channel output disassembly registers plus a disassembly register for the MCU. This network is illustrated in greater detail in figure 2-30 on the following page. Data read from SCM for delivery to an output channel passes through the SRO register. The 60 data bits are then transmitted to an output data buffer register through a static delay network. From the buffer register the data is distributed to the 15 output disassembly registers and the MCU disassembly register.

Output data buffer register

The output data buffer register is a 60 bit clear/enter type register which is cleared and entered with new data at the end of every clock period. The data in the SRO register is transmitted to the output data buffer register during each clock period. The data in the buffer register is then one clock period later than the corresponding data in the SRO register. The output data buffer register distributes this data to the 15 output disassembly registers and the MCU disassembly register through a static data distribution network. The entry of data in the individual disassembly registers is under the control of the associated channel control unit.

Output disassembly register

The output disassembly register is a 60 bit clear/enter type register with gated clock pulse control. There is one such register associated with each output channel and one with the MCU. The data in the register is cleared and new data entered at the end of a clock period in which the output word resume flag is set in the associated channel output control unit. This clock period corresponds to the time at which the data requested by the channel output control unit has arrived at the output data buffer register and is available in the static data distribution network.

A static selection network chooses one 12 bit parcel from the output disassembly register for transmission over the channel output data path to a PPU. This selection is controlled by the output parcel count register in the channel output control unit. The highest order parcel in the disassembly register corresponds to a zero value in the output parcel count register. The lowest order parcel in the disassembly register corresponds to a value of four in the output parcel count register.

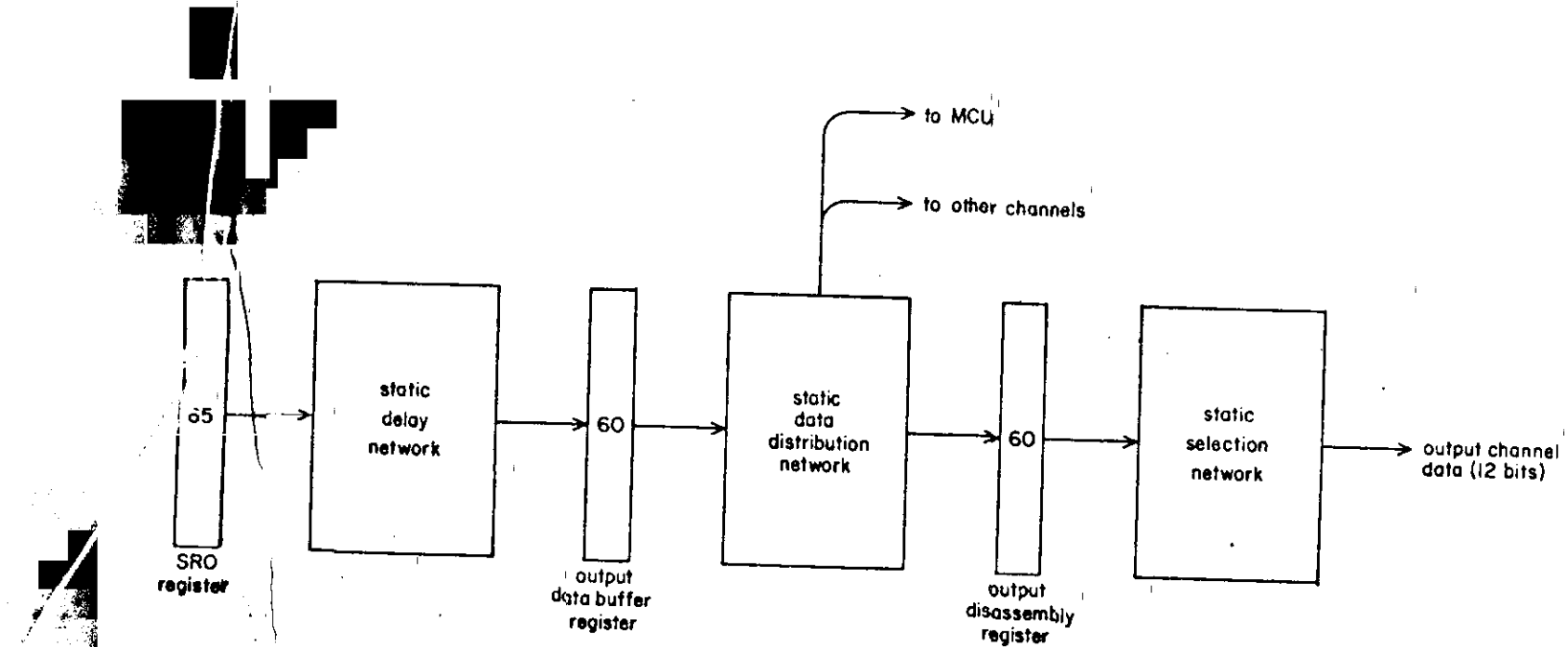


Fig. 2-30 Ouput Data Distribution Network

I/O Word Request Control

The 15 channel input control units and 15 channel output control units each contain a word request flag. These flags are set when the associated control unit requires a SCM storage reference for reading an output word or for storing an input word. Figure 2-31 on the following page illustrates the common control mechanism for processing these requests.

At the left edge of figure 2-31 are the flags and address registers representing the channel input and channel output control units. Only one set of flags and registers are illustrated. There are actually 15 such sets plus a set for the MCU. There is no distinction between input and output devices in the merging mechanism for the request flags. The flags are treated as 32 word request flags each independently requesting the use of SCM.

Priority request flag

There are 32 priority request flags in the I/O word request control. These flags are one bit clear/enter type registers which are cleared at the end of every clock period. There is one priority request flag associated with each input word request flag and another associated with each output word request flag. The priority request flag is cleared and entered with data from the associated word request flag at the end of every clock period. The priority request flag therefore is a copy of the word request flag with a one clock period delay.

Priority resume flag

There are 32 priority resume flags in the I/O word request control. These flags are one bit registers with a separate set and a separate clear input. A flag is set when the associated channel control unit has completed a SCM reference. The flag remains set during the requesting process for the next SCM reference. A priority resume flag is cleared when the associated priority request flag has been recognized by the priority selection network and the request is in process by SCM. These two flags work together to request a SCM reference and then quickly remove the request to free the mechanism for a different channel. In this manner the priority selection network may operate at a higher speed than if communication were required directly with the channel control units.

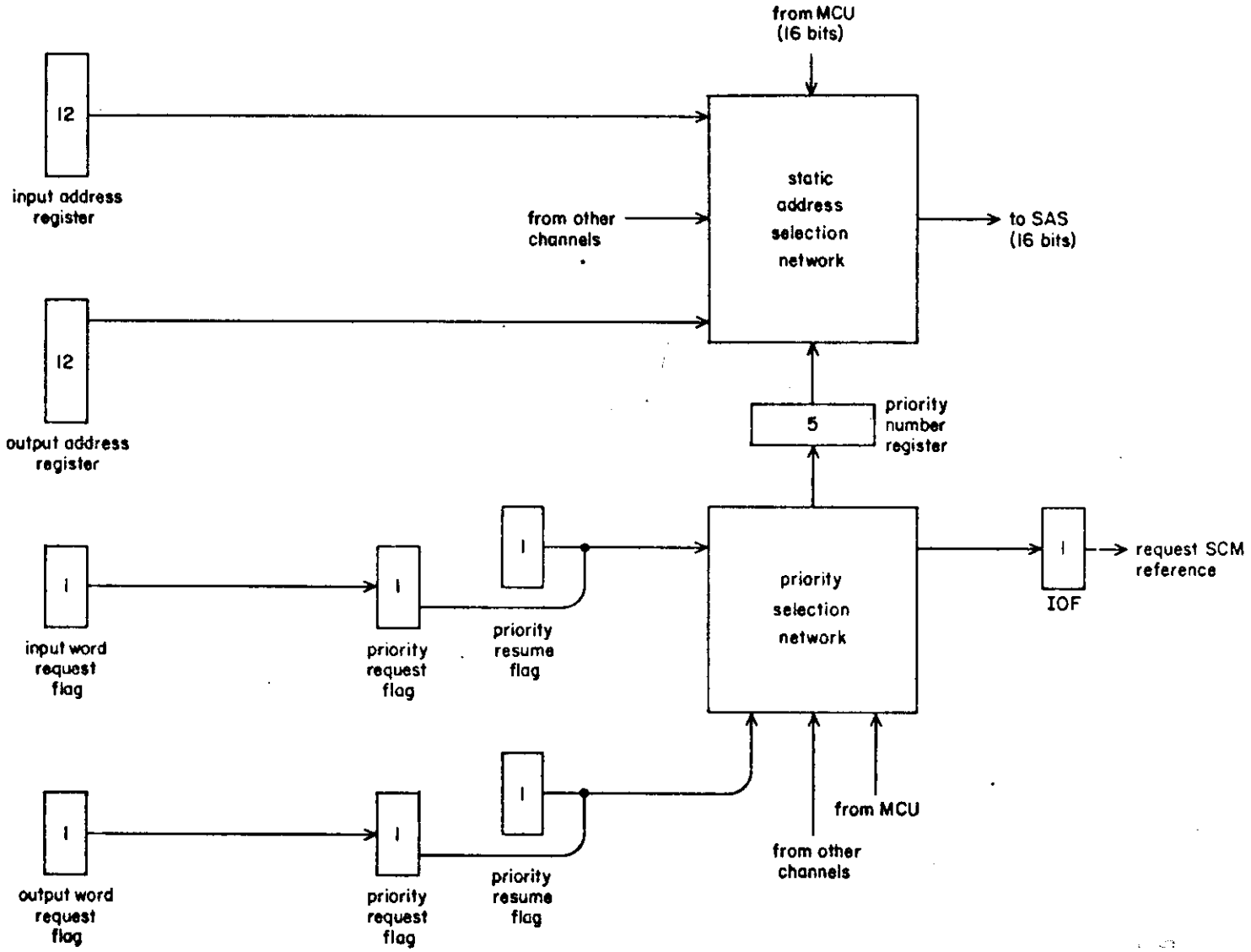


Fig. 2-31 I/O Word Request Control

A priority resume flag is set at the end of a clock period in which the following condition is satisfied.

Condition: no word request flag (input or output)

A priority resume flag is cleared at the end of a clock period in which the following condition is satisfied.

Condition: "accept I/O" & this flag designated in priority number register
or: "dead start"

Input/output flag (IOF)

This flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag may be set by the priority selection network at the end of a clock period in which a priority request flag and a priority resume flag are set for the same channel. The condition for setting the IOF is as follows:

Condition: no "accept I/O" & no XSF & priority request flag 0 & priority resume flag 0
or: no "accept I/O" & no XSF & priority request flag 1 & priority resume flag 1
or: no "accept I/O" & no XSF & priority request flag 2 & priority resume flag 2
or: no "accept I/O" & no XSF & priority request flag 3 & priority resume flag 3

or: no "accept I/O" & no XSF & priority request flag 37 (octal) & priority resume flag 37 (octal)

Priority selection network

The priority selection network illustrated in figure 2-31 is a static network which chooses the highest priority channel request for SCM processing at the end of every clock period. A channel request exists when the priority request flag and the priority resume flag are both set for that channel. If no channel request exists during a particular clock period, the IOF is not set at the end of that period and a zero value is entered in the priority number register. If one channel request exists during a particular clock period, a request priority number for that channel is entered in the priority number register. The IOF will be set at this same time unless a conflict exists with the XSF or the "accept I/O" condition.

If more than one channel request exists during a given clock period, the priority selection network chooses the highest priority channel for processing. The channel priority numbers are listed below with the highest priority first.

Priority 00 - MCU input
Priority 01 - MCU output
Priority 02 - Channel one input
Priority 03 - Channel one output
Priority 04 - Channel two input
Priority 05 - Channel two output
Priority 06 - Channel three input
Priority 07 - Channel three output
Priority 10 - Channel four input
Priority 11 - Channel four output
Priority 12 - Channel five input
Priority 13 - Channel five output
Priority 14 - Channel six input
Priority 15 - Channel six output
Priority 16 - Channel seven input
Priority 17 - Channel seven output
Priority 20 - Channel ten input
Priority 21 - Channel ten output
Priority 22 - Channel eleven input
Priority 23 - Channel eleven output
Priority 24 - Channel twelve input
Priority 25 - Channel twelve output
Priority 26 - Channel thirteen input
Priority 27 - Channel thirteen output
Priority 30 - Channel fourteen input
Priority 31 - Channel fourteen output
Priority 32 - Channel fifteen input
Priority 33 - Channel fifteen output
Priority 34 - Channel sixteen input
Priority 35 - Channel sixteen output
Priority 36 - Channel seventeen input
Priority 37 - Channel seventeen output

Priority number register

This is a five bit clear/enter type register which is cleared at the end of every clock period. This register holds the current highest priority channel request number for processing by the SCM access control. The number held in this register is used by the static address selection network to select the proper input address register value, or output address register value, for transmission to the SAS. This same number is used to clear the priority resume flags and in the I/O word resume control. A new value is entered in this register from the priority selection network at the end of every clock period.

Static address selection network

This network selects the proper input address register value, or output address register value, for transmission to the SAS as illustrated in figure 2-31. Selection of the proper register value is based on the contents of the priority number register. One register value is selected in each clock period and transmitted to the SAS. This address is not entered in the SAS unless the IOF is set and no conflict exists in the SCM access control unit. All addresses are treated as 16 bit quantities. The channel address registers which contain only 12 bits are extended with four higher order zero bits.

I/O Word Resume Control

This portion of the CPU I/O section controls the processing of an I/O word reference from the time of address arrival at SCM to the resume of the channel control. The organization of the I/O word resume control is illustrated in figure 2-32 on the following page.

A sequence of events in the I/O word resume control begins with the setting of the IOF in the I/O word request control. Simultaneously the priority number register is set with the designation of the proper channel and mode for the requested reference. This flag and register value are transmitted through a chain of six bit registers and static networks as illustrated in figure 2-32. The progress of the six bit code through this chain parallels the progress of data through the SWS. The rank designations in this chain correspond to the same ranks in the SWS. There are two major points in the chain at which the control interfaces to other parts of the I/O section. The I/O resume rank D register value selects the proper input data path for data transmitted to SCM. The I/O resume rank H register value selects the proper control path for clearing a word request flag in a channel input control unit or a channel output control unit.

I/O resume rank B register

This is a six bit clear/enter type register with gated clock pulse control. The highest order bit in this register is a copy of the IOF from the I/O word request control unit. The lowest order five bits in this register are copies of the data in the priority number register. The control information stored in this register corresponds to the data stored in rank B of the SWS. This register is cleared and new data entered at the end of a clock period in which the following condition exists.

Condition: NBF
or: NCF

I/O resume rank C register

This is a six bit clear/enter type register with gated clock pulse control. The control information stored in this register corresponds to the data stored in rank C of the SWS. This register is cleared and new data is entered from the I/O resume rank B register at the end of a clock period in which the following condition exists.

Condition: NCF

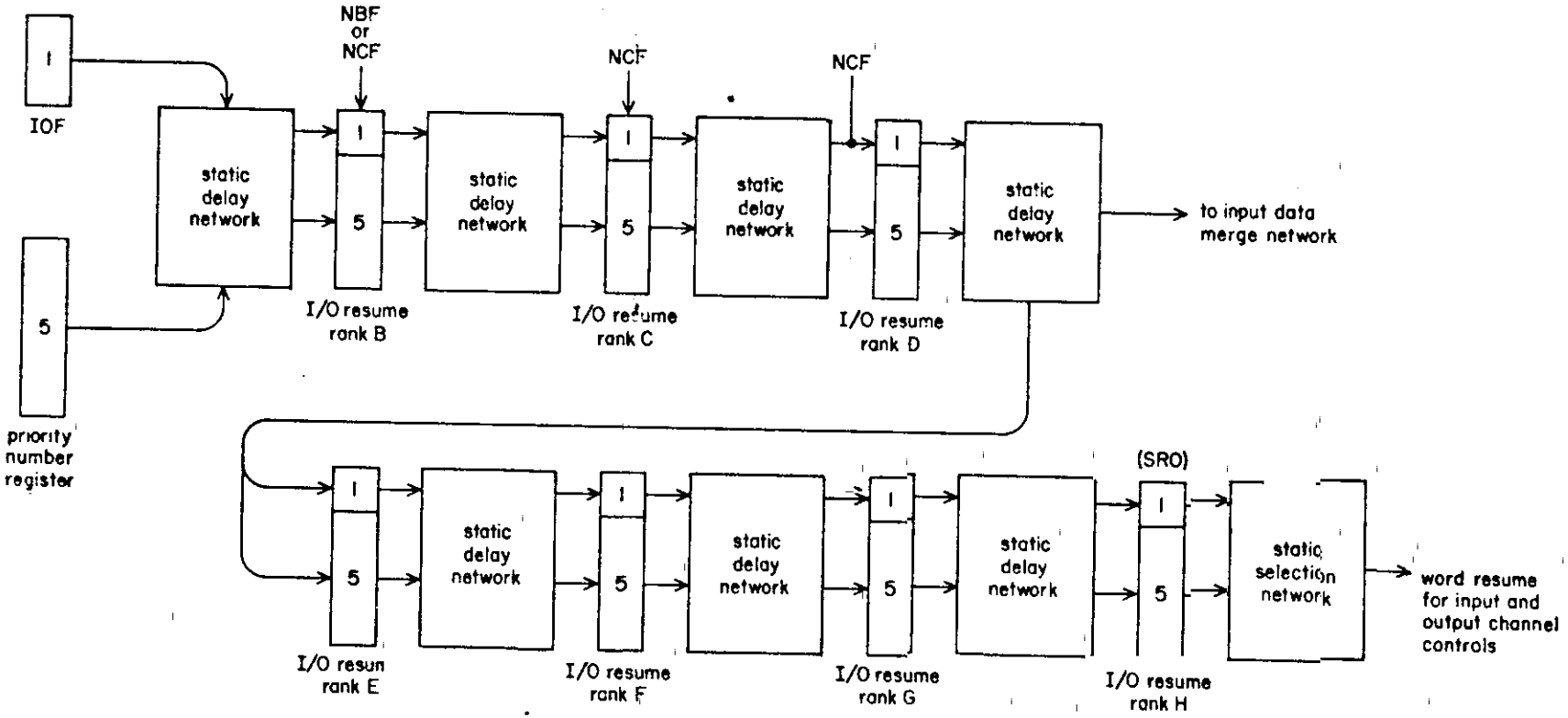


Fig. 2-32 I/O Word Resume Control

I/O resume rank D register

This is a six bit clear/enter type register which is cleared at the end of every clock period. The control information stored in this register corresponds to the data stored in rank D of the SWS. This register has two basic functions. It relays the six bits of control information to the I/O resume rank E register, and it controls the input data merge network for data transmitted to SCM. The input data merge network is illustrated in figure 2-28. There are sixteen sets of 60 bit data paths merged in this network for transmission to the SWS rank E register. The selection of one of these sixteen data paths is controlled by the center four bits of the data in the I/O resume rank D register.

The data in the I/O resume rank D register is cleared at the end of every clock period and new data is entered from the rank C register. The highest order bit in this transmission path is treated in a special manner. This bit corresponds to the IOF for the word request. The bit is set in the rank D register only on the following condition.

Condition: NCF & highest order bit set in rank C register

I/O resume rank E register

This is a six bit clear/enter type register which is cleared at the end of every clock period. The control information stored in this register corresponds to the data stored in the SWS rank E register. The data is cleared in this register, and new data is entered from the rank D register, at the end of every clock period.

I/O resume rank F register

This is a six bit clear/enter type register which is cleared at the end of every clock period. The control information stored in this register corresponds to the data stored in the SWS rank F register. The data is cleared in this register, and new data is entered from the rank E register, at the end of every clock period.

I/O resume rank G register

This is a six bit clear/enter type register which is cleared at the end of every clock period. The control information stored in this register corresponds to the data stored in the SWS rank G register. The data is cleared in this register, and new data is entered from the rank F register, at the end of every clock period.

I/O resume rank H register

This is a six bit clear/enter type register which is cleared and entered with data from the rank G register at the end of every clock period. This is the last rank in the I/O resume chain. The control information in this register corresponds to the data in the SRO register. The highest order bit is set in this register when the data in the SRO register is intended for transmission to an output channel disassembly register. In this case the lowest order five bits in the I/O resume rank H register determine the proper channel control unit for the resume information. If the lowest order bit is zero, an input word resume condition is present for the input channel designated by the next highest order four bits. If the lowest order bit is one, an output word resume condition is present for the output channel designated by the next highest order four bits. In this last case the output word resume flag is set for the appropriate channel output control unit. This flag will then be set at the same time as the data in the SRO register arrives at the output data buffer register illustrated in figure 2-30. The output word resume flag then controls the data transmission from the output data buffer register to the output disassembly register for the appropriate channel.

I/O Interrupt Control

This portion of the CPU I/O section controls the processing of input channel and output channel interrupt requests. These requests are processed in a section of the CPU which is completely separate from the processing of input channel and output channel word requests. The organization of this control section is illustrated in figure 2-33 on the following page.

A sequence of events in the I/O interrupt control begins with the setting of an input interrupt request flag or an output interrupt request flag. One of each of these flags is illustrated at the left edge of figure 2-33. There are 15 such input interrupt request flags and 15 output interrupt request flags in the various channel control units. In addition, there are two interrupt request flags associated with the MCU. These last two flags are treated in the I/O interrupt control as if they were the input and output flags associated with a channel zero.

The 32 independent interrupt request flags are connected to an interrupt selection network as illustrated in figure 2-33. This network senses the presence of an interrupt request and encodes an identifying number in the I/O exchange request register. The following static network then creates an "I/O exchange request" condition which is transmitted to the exchange sequence control unit. This static network also forms a SCM address from the information in the I/O exchange request register. The exchange sequence control then causes the current CPU program to be interrupted with an exchange sequence using the specified SCM address.

A CPU program associated with the referenced exchange package performs the necessary functions for the requesting channel. Concurrently, an "I/O exchange resume" condition is generated in the exchange sequence control unit and is transmitted to the I/O interrupt control where it sets the I/O exchange accepted flag. This flag remains set during the execution of the CPU program for the channel interrupt. The next action in the I/O interrupt control occurs when the CPU program is completed and an 013 instruction is executed to return the CPU to the previous program. The execution of the 013 instruction sets the PXF in the exchange sequence control unit. This flag is transmitted to the I/O interrupt control and begins the process of releasing the interrupt request mechanism.

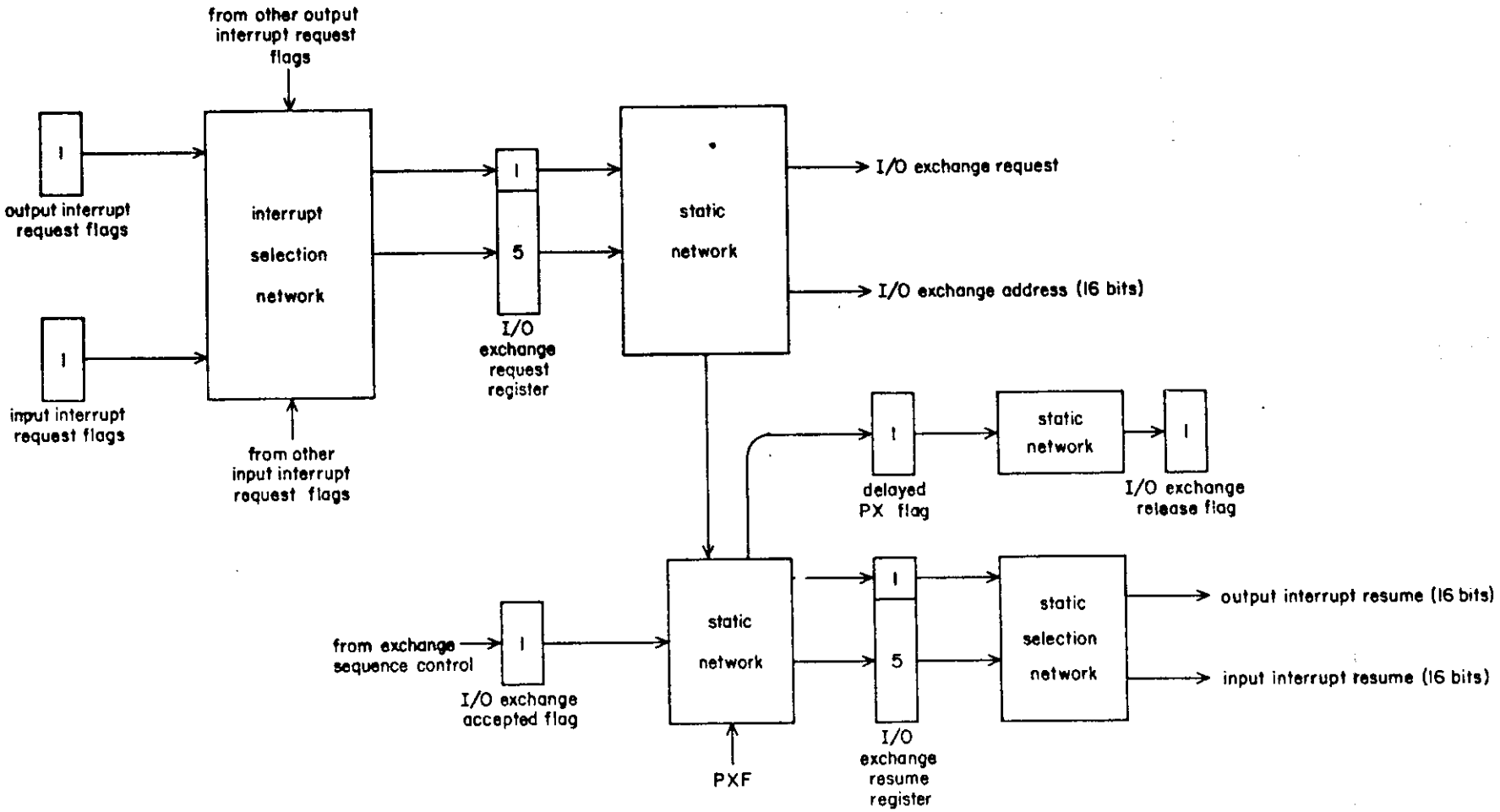


Fig. 2-33 I/O Interrupt Control

The PXF causes the data in the I/O exchange request register to be copied into the I/O exchange resume register. It also sets the delayed PX flag illustrated in figure 2-33. A static network following the I/O exchange resume register translates the number in the register and sends an input interrupt resume, or an output interrupt resume, to the requesting channel control unit. This resume clears the interrupt request flag in the channel control unit and completes the interrupt sequence. Concurrently, the I/O exchange release flag frees the I/O exchange request register for use in the next interrupt request sequence.

Interrupt selection network

This is a static network in the I/O interrupt control unit which senses an interrupt request from a channel control unit and encodes an identifying number in the I/O exchange request register. If more than one input interrupt request flag (or output interrupt request flag) is set in a given clock period, the interrupt selection network encodes the identifying number associated with the flag having the highest priority. The priority numbers for interrupt control are the same as the corresponding numbers for word requests. These numbers are listed on page 2-166.

I/O exchange request register

This is a six bit clear/enter type register with gated clock pulse control. This register is cleared and entered with data from the interrupt selection network at the end of a clock period in which the following condition exists.

Condition: I/O exchange release flag
or: no tag bit in I/O exchange request register

This register is cleared and entered with data from the interrupt selection network at the end of each clock period until an interrupt request is encoded by the interrupt selection network. This encoded value plus a tag bit is then held in the register until the I/O exchange release flag is set.

A "dead start" condition for the CPU forces the contents of the I/O exchange request register to an interrupt for input channel 0. The final action for the CPU on release from "dead start" will be an exchange sequence to SCM address zero.

"I/O exchange request" condition

This condition is generated in the I/O interrupt control unit when an interrupt request has been encoded into the I/O exchange request register. This condition is present whenever the highest order bit in the I/O exchange request register is set. This bit is a tag bit which is set whenever an input interrupt request flag or an output interrupt request flag has been processed by the interrupt selection network. The bit remains set until the I/O exchange release flag causes a new entry into the I/O exchange request register.

I/O exchange address

A 16 bit I/O exchange address is generated in the I/O interrupt control unit in a static network following the I/O exchange request register. This address is generated from the lowest order five bits in the I/O exchange request register. The highest order seven bits of the 16 bit address are always zero. The lowest order nine bits are formed by shifting the five bit encoded priority number in the I/O exchange request register left by four bit positions. This address then specifies the proper exchange package location in SCM as illustrated in figure 1-4 in part one of this manual.

I/O exchange accepted flag

This flag is a one bit register with a separate set and a separate clear input. The flag is set at the end of a clock period in which the following condition is present.

Condition: "I/O exchange resume"

This flag is cleared at the end of a clock period in which the following condition exists.

Condition: I/O exchange release flag
or: no tag bit in I/O exchange request register

This flag remains set from the time the exchange sequence control unit begins the initial exchange for the interrupt request until the CPU program associated with the interrupt request has been completed and an exchange exit instruction is executed.

Delayed PX flag

This flag is a one bit clear/enter type register which is c ared at the end of every clock period. The flag is reset at the e of a clock period in which the PXF is set. This flag is theref a copy of the PXF with a one clock period delay.

I/O exchange release flag

This flag is a one bit clear/enter type register which is red at the end of every clock period. This flag is set at the e f a clock period in which the following condition is present.

Condition: Delayed PX flag & I/O exchange accepted flag

I/O exchange resume register

This is a six bit clear/enter type register which is cleared at the end of every clock period. This register is entered with a copy of the contents of the I/O exchange request register at the end of a clock period in which the following condition is present.

Condition: PXF & I/O exchange accepted flag

This register is entered at the same time as the delayed PX flag is set. A static network following this register translates the lowest order five bits in the register and forms an interrupt resume for a specific input or output channel control unit. The interrupt request flag in that channel control unit will then be cleared at the same time as the I/O exchange release flag is set in the I/O interrupt control.

3 CPU INSTRUCTIONS

PART 3: CPU INSTRUCTIONS

Introduction

This section of the reference manual is devoted to describing the CPU instructions in considerable detail. Each instruction is described separately as to what it does, briefly how it does it, how long it takes to do it, and what happens if unusual or special situations arise. Extensive use is made of three letter abbreviations and special terms which are defined in part two of this manual. These terms and abbreviations are listed in the index at the back of the manual, together with the page reference in part two where each term is defined.

The CPU instructions tend to fall into two distinct categories: those causing computation, and those causing storage references or program branching. The CPU instructions causing computation are generally executed in a fixed amount of time after they have issued from the CIW register. A string of such instructions may issue in consecutive clock periods and perform a large amount of computation in a very short time. More generally, each instruction waits for one or more clock periods before it issues from the CIW register because one or both of the operands required for that instruction have not yet arrived at the operating registers. Careful coding of critical program loops can therefore result in substantial improvements in execution time. The detail timing information provided in this part of the reference manual is intended to allow a complete analysis of these situations where the programing effort is warranted.

Instructions involving storage references for operands or program branching cannot be precisely timed. The relatively random SCM storage bank conflicts resulting from I/O section storage references reduce this area of programing to a statistical problem. Program branching within the instruction stack causes no storage references, and small program loops can therefore be precisely timed for this case.

00xxx Error exit

This instruction format is treated as an error condition and, if executed, will set the program range condition flag in the PSD register. This condition flag will then generate an error exit request which will cause an exchange jump to address (EEA). In this case all instructions which have issued prior to this instruction will be run to completion. Any instructions following this instruction in the current instruction word will not be executed. When all operands have arrived at the operating registers as a result of previously issued instructions, an exchange jump will occur to the exchange package designated by (EEA).

The i, j, and k designators in this instruction are ignored. The program address stored in the exchange package on the terminating exchange jump is advanced one count from the address of the current instruction word. This is true no matter which parcel of the current instruction word contains the error exit instruction.

This instruction format is not intended for use in normal program code. The program range condition flag is set in the PSD register to indicate that the program has jumped to an area of the SCM field which may be in range but is not valid program code. This should occur when an incorrectly coded program jumps into an unused area of the SCM field or into a data field. The program range condition flag is also set on the condition of a jump to address zero, or a jump beyond the SCM field length. These conditions can be determined by the system monitor program on the basis of the register contents in the exchange package. The existence of an error exit condition resulting from execution of this instruction format may thus be deduced by the monitor program.

Special situations

A special situation may occur when a program is terminated with an error exit format and a previously issued instruction stores a result operand in SCM. The error exit is treated as a SCM range error which blocks a write operation in SCM as soon as the error is detected. A legitimate SCM write operation may be blocked by the error condition even though the instruction causing the write issues substantially before the error exit. The timing in this case will depend on the SCM bank conflicts which may have occurred.

0100x	xxxxx

Return jump

This is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction writes a special word into the SCM field at relative address K. The instruction stack is cleared, and the current program sequence is then terminated by a jump to address K + 1 in the SCM field. The word stored in SCM contains a jump instruction which, when executed, will cause an unconditional jump to the address of this return jump instruction plus one.

This instruction is intended to call a subroutine and insert execution of this subroutine between execution of the current instruction word and the following instruction word. Instructions appearing after the return jump instruction in the current instruction word will not be executed. The called subroutine exit must be at address K in the SCM field. The called subroutine entrance address must be K + 1 in the SCM field.

This instruction stores a full 60 bit word at address K in the SCM field. The upper half of this word contains an unconditional jump instruction (0400) with an address which is equal to the current program address plus one. The lower half of the stored word is all zeros. The octal digits in the stored word then appear as shown below with the xxx field indicating the location of the current program address plus one.

K	0400x xxxxx 00000 00000	subroutine exit
K + 1	yyyyy yyyyy yyyyy yyyyy	subroutine entrance

Execution time

The minimum execution time for a return jump instruction is 13 clock periods. The return jump sequence begins as soon as the instruction enters the upper parcel of the CIW register. The sequence does not wait for the completion of previously issued instructions. The following is a chronological listing of events in the return jump sequence for the case of minimum execution time.

CP00 010 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Set GJF.

CP01 010 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit (P) to the RJX register.
Transmit K to the P register.
Set RJF.
Clear GJF.

CP02 010 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Void (IWS). Clear (IAS).
Transmit (P) + (RAS) to the IFA register.
Set SXF.
Clear RJF.

010 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit (IFA) to the SAS. Tag for store exit address.
Transmit (RJX) to the SWS.
Advance (P).
Set JCF.
Clear SXF.

CP04 010 instruction in the upper parcel of the CIW register.
Instruction issues.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
Set JOF.
Set OSF.
Transmit (P) + (RAS) to the IFA register.
Set F1F, F2F.
Clear JCF.

No instruction in the upper parcel of the CIW register.
Transmit (P) to the NSA register.
Set M1F, M2F.
The OSF remains set.
Transmit (IFA) to the SAS. Tag for read to IWS.
Advance (IFA).
Clear JOF.
Clear F2F.

CP06 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Transmit (IFA) to the SAS. Tag for read to IWS.
Advance (IFA).
First fetch address leaves the SAS for a SCM bank.
Clear F1F.

CP07 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
First fetch memory read/write cycle begins in a SCM bank.
Second fetch address leaves the SAS for a SCM bank.

CP08 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Second fetch memory read/write cycle begins in a SCM bank.

CP09 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.

CP10 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
First fetch instruction word reads to SCM bank operand register

CP11 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Transmit the first fetch instruction word to the IWS.
Transmit (NSA) to the IAS.
Shift the IWS and the IAS one word position.
Advance (NSA).
Second fetch instruction word reads to SCM bank operand register.
Clear M2F.

No instruction in the upper parcel of the CIW register.
Coincidence in the IAS.
Read a valid instruction word from the IWS to the CIW register.
Clear the OSF.
Transmit the second fetch instruction word to the IWS.
Transmit (NSA) to the IAS.
Shift the IWS and the IAS one word position.
Advance (NSA).
Clear M1F.

Next instruction in the upper parcel of the CIW register
Instruction may issue.

Execution delays

A delay will occur in the execution of the return jump sequence if M1F is set during CP02 in the timing sequence listed above. This condition will exist if the instruction stack control unit has requested one or more fetch instruction words which have not arrived at the instruction stack by CP02. This is likely to be the case if the sequence of instructions prior to the return jump instruction has been straight line coding for several instruction words. The SXF will not set until the M1F has been cleared indicating that all instruction words requested for the instruction stack have arrived. Execution of the commands indicated in CP02 of the listing above will be delayed in this case by the number of clock periods required to complete the instruction fetches and clear the M1F. All subsequent commands in later clock periods will be delayed by the same amount.

A delay will occur in the execution of the return jump sequence if a backup condition exists in the SAS during CP03 in the timing sequence listed above. This condition may exist if SCM storage references unrelated to the return jump sequence caused a storage bank conflict just prior to CP03. Execution of the CP03 commands will be delayed in this case until the NBF sets indicating the SAS backup situation has been resolved. All subsequent commands in later clock periods will be delayed by the same amount.

A delay will occur in the execution of the return jump sequence if (IFA) is not able to be transmitted to the SAS during CP05 because of a conflict with an I/O section word request or because of a SAS backup condition. The I/O section has priority over instruction stack fetch requests in the SCM access control unit. If such a request happens to occur during CP05 of the timing sequence listed above, the instruction stack fetch is delayed by one clock period. A backup condition can exist in the SAS during CP05 due to a SCM bank conflict between the store exit reference of the return jump sequence and an unrelated SCM reference, or between two unrelated SCM references. In any of these cases the remainder of the return jump sequence is delayed by the number of clock periods required to resolve the conflict or backup condition.

A delay will occur in the execution of the return jump sequence if a SCM bank conflict exists during CP06 of the timing sequence listed above. This situation will occur if the address of the first fetch instruction word requires a SCM bank which is busy with a prior reference. The arrival of the first fetch instruction word at the IWS will be delayed in this case by the number of clock periods required for the SCM bank to complete the read/write cycle required in the previous reference.

Special situations

Designator j not zero

The j designator in the return jump instruction is normally zero This designator is ignored, however, in the execution of the instruction, and a non-zero value will have no effect on the results.

Last parcel

The return jump instruction requires two parcels of an instruction word for normal use. If a return jump instruction begins in the first, second, or third parcel of an instruction word the following parcel completes the instruction. If a return jump instruction begins in the last parcel of an instruction word it will not be continued in the following word. In this case the instruction will be executed as if there were a fifth parcel in the instruction word and this parcel contained all zeros.

Jump out of range

A special situation exists if the value of K in a return jump instruction is greater than the SCM field length. In this case the instruction is executed with the store of the exit word in SCM inhibited. The program address is altered to the value K and advanced by one count in a normal manner. The program range condition flag is set in the PSD register to indicate the jump out of range. The program sequence is then terminated with an exchange jump to (EEA). The resulting exchange package will contain a program address equal to K + 1, and a bit set in the PSD area corresponding to the program range condition flag.

Jump to zero

A special situation exists if the value of K in the return jump instruction is zero. In this case the instruction is executed in a normal manner, and the exit word is stored at address zero in the SCM field. In the process of executing the instruction (P) is momentarily set to zero. This is sensed as an error condition, and the program range condition flag is set in the PSD register. As a result, the program sequence will be terminated at the completion of the return jump instruction with an exchange jump to (EEA). The return jump instruction will have advanced the program address one count so that the exchange package will indicate a program address of one rather than zero.

Jump to breakpoint address

A special situation exists if the value of K in the return jump instruction is equal to (BPA). In the process of executing the instruction (P) will momentarily be set equal to (BPA). This will be detected as a breakpoint condition, and the breakpoint condition flag will set in the PSD register. The return jump instruction will advance (P) one count in the process of completing execution. This final value of (P) will appear in the exchange package when the breakpoint interrupt occurs.

Error condition during execution

A number of error conditions may occur during the execution of a return jump sequence. Some possible conditions are arithmetic errors due to previously issued instructions and parity errors in SCM or LCM. If any error conditions occur during the return jump sequence, the proper flags are set in the PSD register and the return jump instruction is executed to completion in a normal manner. The program sequence is then terminated with an exchange jump to (EEA). The resulting exchange package will contain a program address equal to K + 1 from the return jump instruction and one or more error flags set in the PSD area.

I/O interrupt during execution

An I/O section interrupt request may occur during the execution of a return jump sequence. In such a case the return jump instruction is completed, and an exchange jump to the proper I/O channel exchange package occurs with the program address equal to K + 1 from the return jump instruction.

011jx	xxxxx

Block copy LCM to SCM

This is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction reads a sequence of 60 bit words from consecutive addresses in LCM and copies them into a block of consecutive addresses in SCM. The block of words begins at address (X0) in the LCM field. The words are stored in the SCM field beginning at address (A0). The number of words to be copied is determined by the sum of K + (Bj).

This instruction is intended to move a quantity of data from the large core memory into the small core memory as quickly as possible. All other activity in the CPU, with the exception of I/O word requests, is stopped during this block transfer of data. All instructions which have issued prior to this instruction are executed to completion. No further instructions are issued until this block transfer is nearly completed. As a result of these restrictions the data flow from LCM to SCM can proceed at the rate of one 60 bit word each clock period. When an I/O section word request for SCM occurs during this transfer, the data flow is interrupted for one clock period. The I/O word address is inserted in the stream of addresses to the SAS, and the addresses for the block transfer are resumed with a one clock period delay.

The maximum number of words which can be copied from LCM to SCM with this instruction is determined only by the size of the SCM field and LCM field. Any block size from a one word block to a field length block can be moved as a unit. The length of the block is determined by adding the quantity K from the instruction to the contents of register Bj. Either quantity may be used to increment, or decrement, the other. The addition is performed in an 18 bit ones complement mode. The resultant sum is treated as an 18 bit positive integer. A zero result will cause this instruction to be executed as a pass instruction.

Three of the parameters for this instruction reside in operating registers (A0, X0, Bj). The contents of these registers are not altered by the execution of this instruction.

Execution time

This instruction remains in the CIW register until the block copy sequence has progressed to the point where all SCM addresses have been delivered to the SAS. This prevents issue of this instruction, and therefore all following instructions, until near the end of the instruction execution. When this instruction issues, a LCM busy flag is set which prevents further LCM references until the block copy has been entirely completed. The program sequence will continue and the following instructions will be executed unless they make reference to large core memory.

The minimum time to issue this instruction for a block of N words is N + 13 clock periods. The minimum time to execute this instruction to completion and clear the LCM busy flag is N + 15 clock periods. A subsequent LCM (011, 012, 014, 015) instruction may then begin execution N + 15 clock periods after beginning execution of this instruction. Any other type of instruction may begin execution N + 13 clock periods after beginning execution of this instruction.

The LCM block access control unit initiates a read/write cycle in three LCM banks at the beginning of this instruction sequence. The first LCM bank will produce a block of eight words which will contain the first LCM word requested in the current instruction. The second and third LCM banks will produce the next consecutive block of 16 words. Additional LCM bank read/write cycles are initiated when required as the transfer progresses. A read/write cycle is initiated in bank n + 3 when the last word is read from bank n. This procedure assures the uninterrupted flow of data as the addressing crosses LCM bank boundaries.

The following is a chronological listing of events in a three word block copy in which no conflicts or delays occur.

011 instruction in the upper parcel of the CIW register.
Instruction does not issue.
All operating registers free.
F1F not set.
LCM busy flag not set.
Transmit (X0) + (RAL) to LCM word address register.
Set go LCM block flag.

011 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit K + (Bj) to LCM block counter.

011 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit first bank address to LCM bank address register.
Transmit (A0) + (RAS) to BAK register.

011 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Begin read/write cycle in first LCM bank.
Transmit second bank address to LCM bank address register.

011 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Begin read/write cycle in second LCM bank.
Transmit third bank address to LCM bank address register.

CP05 011 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Begin read/write cycle in third LCM bank.

CP06 011 instruction in the upper parcel of the CIW register.
Instruction does not issue.

CP11 011 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Set go block copy flag.

CP12 011 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit (BAK) to the SAS. Tag for LCM to SCM.
Advance (BAK).
Reduce LCM block count.

CP13 011 instruction in the upper parcel of the CIW register.
Instruction does not issue.
First LCM bank reads 8 words to LCM bank operand register.
First SCM address leaves SAS for SCM bank address register.
Transmit (BAK) to the SAS. Tag for LCM to SCM.
Advance (BAK).
Reduce LCM block count.

CP14 011 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Second LCM bank reads 8 words to LCM bank operand register.
Begin read/write cycle for first word in SCM bank.
Second SCM address leaves SAS for SCM bank address register
Transmit (BAK) to the SAS. Tag for LCM to SCM.
Advance (BAK).
Reduce LCM block count.
Clear go block copy flag.
Advance LCM word address.

CP15 011 instruction in the upper parcel of the CIW register.
Instruction issues.
Third LCM bank reads 8 words to LCM bank operand register.
Transmit next instruction to upper parcel of the CIW register.
First 60 bit word arrives at LCM read register.
Begin read/write cycle for second word in SCM bank.
Third SCM address leaves SAS for SCM bank address register.
Advance LCM word address.
Set LCM busy flag.

CP16 Next instruction in the upper parcel of the CIW register
Instruction may issue.
Transmit first word from LCM read register to SWS.
Second 60 bit word arrives at LCM read register.
Begin read/write cycle for third word in SCM bank.
Advance LCM word address.
LCM busy flag remains set.

CP17 First SCM reference reads to SCM bank operand register.
Transmit second word from LCM read register to SWS.
Third 60 bit word arrives at LCM read register.
Clear LCM busy flag.

CP18 Transmit first word from SWS to SCM bank operand register.
Second SCM reference reads to SCM bank operand register.
Transmit third word from LCM read register to SWS.

CP19 Transmit second word from SWS to SCM bank operand register
Third SCM reference reads to SCM bank operand register.

CP20 Transmit third word from SWS to SCM bank operand register.

Execution delays

The execution of this instruction will not begin until the three conditions listed below have been satisfied.

All operating registers must be free: This implies that all previously issued instructions have delivered their results to the operating registers. This will normally not be the case in the clock period in which this instruction first appears in the CIW register. A delay of two or three clock periods would be normal for this condition.

The FIF must not be set: This will be the case unless a SCM bank conflict has caused a SAS backup condition which prevented an instruction fetch address from leaving the IFA register. This condition is not likely to occur often enough to cause a significant average delay.

The LCM busy flag must not be set: This will be true unless another LCM (011, 012, 014, 015) instruction has just preceded this instruction.

A delay will occur at CP01 in the sequence above if any LCM bank is in the process of completing a read/write cycle from a previous LCM reference. A LCM read/write cycle requires 64 clock periods for completion. A significant portion of this time could appear as a delay in the execution of this instruction if a LCM instruction has been recently executed.

A delay will occur during the block transfer of data from LCM to SCM whenever an I/O section word request is made. A minimum delay of one clock period is required to enter the I/O word address in the address stream to the SAS. An additional delay will occur if the I/O reference causes a bank conflict in SCM with one of the block copy references. The block copy references to SCM cannot cause a bank conflict among themselves since the addresses in the block are sequential and the SCM read/write cycle is less (10 clock periods) than the number of banks (32) available. The probability of a bank conflict between a random I/O address and one of the block copy addresses preceding or following it is, however, rather high (18/32).

The average delay caused by each I/O word request during the block copy is 3.81 clock periods. The gross effect of light I/O volume (6 million bits/sec) is a 1 per cent increase in the block copy time over no I/O activity. The gross effect of heavy I/O volume (60 million bits/sec) is a 10 per cent increase in block copy time.

Special situations

(X0) negative
(X0) greater than 19 significant bits

The lowest order 19 bits of (X0) are used to determine the initial address in the LCM field for the block copy. The higher order bits are ignored. If (X0) is negative the lowest order 19 bits are masked out and treated as a positive integer.

LCM out of range

A test against LCM field length is made at the beginning of the block copy sequence. The length of the block is determined by adding the quantity K to (Bj) in an 18 bit ones complement mode. The resulting sum is treated as an 18 bit positive integer. This integer is added to the lowest order 19 bits of (X0), also treated as a positive integer. The resulting sum is compared with (FLL). If the resulting sum is greater than (FLL), indicating that the block copy will go beyond the assigned LCM field, the block copy is not executed. In this case the LCM block range condition flag is set in the PSD register, and the block copy instruction is issued as a pass with a four clock period execution time. The exchange jump to (EEA) resulting from setting the LCM block range condition flag will not occur before execution of the next program instruction word unless a delay is introduced by subsequent instructions in the current instruction word.

SCM out of range

A test against SCM field length is made at the beginning of the block copy sequence. The length of the block is determined by adding the quantity K to (Bj) in an 18 bit ones complement mode. The resulting sum is treated as an 18 bit positive integer. This integer is added to (A0), also treated as an 18 bit positive integer. The resulting sum is compared with (FLS). If the resulting sum is greater than (FLS), indicating that the block copy will go beyond the assigned SCM field, the block copy is not executed. In this case the SCM block range condition flag is set in the PSD register, and the block copy instruction is issued as a pass with a four clock period execution time. The exchange jump to (EEA) resulting from setting the SCM block range condition flag will not occur before execution of the next program instruction word unless a delay is introduced by subsequent instructions in the current instruction word.

Block length negative

The length of the block is determined by adding the quantity K from the instruction to the contents of register Bj. The addition is performed in an 18 bit ones complement mode. The resultant sum is treated as an 18 bit positive integer. A negative result will therefore appear as a large positive integer. In this case the SCM block range condition flag, and possibly the LCM block range condition flag, will set in the PSD register, indicating too large a block for the assigned fields. The block copy instruction will issue as a pass with a four clock period execution time. The exchange jump to (EEA) resulting from setting the SCM block range condition flag will not occur before execution of the next program instruction word unless a delay is introduced by subsequent instructions in the current instruction word.

Block length zero

A zero block length is treated as a normal situation. No error flags are set. The block copy instruction is executed as a pass with a four clock period execution time.

LCM words already in bank operand register

The LCM words required for the block copy instruction may already be in one of the LCM bank operand registers from the execution of a previous instruction. This situation is not sensed. The words in the LCM bank operand register are discarded and are reread from the LCM bank.

Last parcel

The block copy instruction requires two parcels of an instruction word for normal use. If this instruction begins in the first, second, or third parcel of an instruction word the following parcel completes the instruction. If a block copy instruction begins in the last parcel of an instruction word it will not be continued in the following word. In this case the instruction will be executed as if there were a fifth parcel in the instruction word and this parcel contained all zeros.

Error condition during execution

A LCM or SCM parity error may occur during the execution of a block copy instruction. An arithmetic error from a previous instruction may also occur during the beginning of the block copy sequence. If any error conditions occur, the proper flags are set in the PSD register and the block copy instruction is executed to completion. There are no error conditions which will interrupt the instruction before completion.

I/O interrupt during execution

An I/O section interrupt request may occur during the execution of a block copy instruction. In this case the interrupt request is not honored until the block copy instruction has been completed and any subsequent instructions in the current instruction word have been completed.

012jx	xxxxx

Block copy SCM to LCM

This is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction reads a sequence of 60 bit words from consecutive addresses in SCM and copies them into a block of consecutive addresses in LCM. The block of words begins at address (A0) in the SCM field. The words are stored in the LCM field beginning at address (X0). The number of words to be copied is determined by the sum of K + (Bj).

This instruction is intended to move a quantity of data from the small core memory into the large core memory as quickly as possible. All other activity in the CPU, with the exception of I/O word requests, is stopped during this block transfer of data. All instructions which have issued prior to this instruction are executed to completion. No further instructions are issued until this block transfer is nearly completed. As a result of these restrictions the data flow from SCM to LCM can proceed at the rate of one 60 bit word each clock period. When an I/O section word request for SCM occurs during this transfer, the data flow is interrupted for one clock period. The I/O word address is inserted in the stream of addresses to the SAS, and the addresses for the block transfer are resumed with a one clock period delay.

The maximum number of words which can be copied from SCM to LCM with this instruction is determined only by the size of the SCM field and LCM field. Any block size from a one word block to a field length block can be moved as a unit. The length of the block is determined by adding the quantity K from the instruction to the contents of register Bj. Either quantity may be used to increment, or decrement, the other. The addition is performed in an 18 bit ones complement mode. The resultant sum is treated as an 18 bit positive integer. A zero result will cause this instruction to be executed as a pass instruction.

Three of the parameters for this instruction reside in operating registers (A0, X0, Bj). The contents of these registers are not altered by the execution of this instruction.

Execution time

This instruction remains in the CIW register until the block copy sequence has progressed to the point where all SCM addresses have been delivered to the SAS. This prevents issue of this instruction, and therefore all following instructions, until near the end of the instruction execution. When this instruction issues, a LCM busy flag is set which prevents further LCM references until the block copy has been entirely completed. The program sequence will continue and the following instructions will be executed unless they make reference to large core memory.

The minimum time to issue this instruction for a block of N words is N + 4 clock periods. The minimum time to execute this instruction to completion and clear the LCM busy flag is N + 11 clock periods. A subsequent LCM (011, 012, 014, 015) instruction may then begin execution N + 11 clock periods after beginning execution of this instruction. Any other type of instruction may begin execution N + 4 clock periods after beginning execution of this instruction.

The LCM block access control does not initiate a LCM bank read/write cycle until all words destined for the first bank have arrived at the corresponding LCM bank operand register. When the last word arrives at the LCM bank operand register, a LCM bank read/write cycle is initiated. As the block copy addressing crosses each bank boundary a LCM bank read/write cycle is initiated for the completed LCM bank entries. At the end of the block copy sequence a LCM bank read/write cycle is initiated for the last LCM bank entered. This procedure assures an uninterrupted flow of data during the block copy sequence.

The following is a chronological listing of events in a three word block copy in which no conflicts or delays occur.

CP00 012 instruction in the upper parcel of the CIW register
Instruction does not issue.
All operating registers free.
FIF not set.
LCM busy flag not set.
Transmit (X0) + (RAL) to LCM word address register.
Set go LCM block flag.

012 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit K + (Bj) to LCM block counter.

012 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit (A0) + (RAS) to BAK register.
Set go block copy flag.

012 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit (BAK) to the SAS. Tag for SCM to LCM.
Advance (BAK).
Reduce LCM block count.

012 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit (BAK) to the SAS. Tag for SCM to LCM.
Advance (BAK).
Reduce LCM block count.
First SCM address leaves SAS for SCM bank address register.

CP05 012 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit (BAK) to the SAS. Tag for SCM to LCM.
Advance (BAK).
Reduce LCM block count.
Begin read/write cycle for first word in SCM bank.
Second SCM address leaves SAS for SCM bank address register.
Clear go block copy flag.

CP06 012 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit next instruction to upper parcel of the CIW register.
Begin read/write cycle for second word in SCM bank.
Third SCM address leaves SAS for SCM bank address register.
Set LCM busy flag.

CP07 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Begin read/write cycle for third word in SCM bank.
LCM busy flag remains set.

CP08 First word reads from SCM bank to SCM bank operand register.
LCM busy flag remains set.

CP09 First word leaves SCM bank operand register for SRO register.
Second word reads from SCM bank to SCM bank operand register.
LCM busy flag remains set.

CP10 Transmit first word from SRO register to LCM write register.
Second word leaves SCM bank operand register for SRO register.
Third word reads from SCM bank to SCM bank operand register.
LCM busy flag remains set.

CP11 First word leaves LCM write register for bank operand register.
Transmit second word from SRO register to LCM write register.
Third word leaves SCM bank operand register for SRO register.
Advance LCM word address.
LCM busy flag remains set.

First word arrives at LCM bank operand register.
Second word leaves LCM write register for bank operand register.
Transmit third word from SRO register to LCM write register.
Advance LCM word address.
LCM busy flag remains set.

Second word arrives at LCM bank operand register.
Third word leaves LCM write register for bank operand register.
Advance LCM word address.
Clear LCM busy flag.

Third word arrives at LCM bank operand register.

Execution delays

The execution of this instruction will not begin until the three conditions listed below have been satisfied.

All operating registers must be free: This implies that all previously issued instructions have delivered their results to the operating registers. This will normally not be the case in the clock period in which this instruction first appears in the CIW register. A delay of two or three clock periods would be normal for this condition.

The F1F must not be set: This will be the case unless a SCM bank conflict has caused a SAS backup condition which prevented an instruction fetch address from leaving the IFA register. This condition is not likely to occur often enough to cause a significant average delay.

The LCM busy flag must not be set: This will be true unless another LCM (011, 012, 014, 015) instruction has just preceded this instruction.

A delay will occur at CP01 in the sequence above if any LCM bank is in the process of completing a read/write cycle from a previous LCM reference. A LCM read/write cycle requires 64 clock periods for completion. A significant portion of this time could appear as a delay in the execution of this instruction if a LCM instruction has been recently executed.

A delay will occur during the block transfer of data from SCM to LCM whenever an I/O section word request is made. A minimum delay of one clock period is required to enter the I/O word address in the address stream to the SAS. An additional delay will occur if the I/O reference causes a bank conflict in SCM with one of the block copy references. The block copy references to SCM cannot cause a bank conflict among themselves since the addresses in the block are sequential and the SCM read/write cycle is less (10 clock periods) than the number of banks (32) available. The probability of a bank conflict between a random I/O address and one of the block copy addresses preceding or following it is, however, rather high (18/32). The average delay caused by each I/O word request during the block copy is 3.81 clock periods. The gross effect of light I/O volume (6 million bits/sec) is a 1 per cent increase in the block copy time over no I/O activity. The gross effect of heavy I/O volume (60 million bits/sec) is a 10 per cent increase in block copy time.

Special situations

(X0) negative
(X0) greater than 19 significant bits

The lowest order 19 bits of (X0) are used to determine the initial address in the LCM field for the block copy. The higher order bits are ignored. If (X0) is negative the lowest order 19 bits are masked out and treated as a positive integer.

LCM out of range

A test against LCM field length is made at the beginning of the block copy sequence. The length of the block is determined by adding the quantity K to (Bj) in an 18 bit ones complement mode. The resulting sum is treated as an 18 bit positive integer. This integer is added to the lowest order 19 bits of (X0), also treated as a positive integer. The resulting sum is compared with (FLL). If the resulting sum is greater than (FLL), indicating that the block copy will go beyond the assigned LCM field, the block copy is not executed. In this case the LCM block range condition flag is set in the PSD register and the block copy instruction is issued as a pass. The exchange jump to (EEA) resulting from setting the LCM block range condition flag will not occur before execution of the next program instruction word unless a delay is introduced by subsequent instructions in the current instruction word.

SCM out of range

A test against SCM field length is made at the beginning of the block copy sequence. The length of the block is determined by adding the quantity K to (Bj) in an 18 bit ones complement mode. The resulting sum is treated as an 18 bit positive integer. This integer is added to (A0), also treated as an 18 bit positive integer. The resulting sum is compared with (FLS). If the resulting sum is greater than (FLS), indicating that the block copy will go beyond the assigned SCM field, the block copy is not executed. In this case the SCM block range condition flag is set in the PSD register and the block copy instruction is issued as a pass. The exchange jump to (EEA) resulting from setting the SCM block range condition flag will not occur before execution of the next program instruction word unless a delay is introduced by subsequent instructions in the current instruction word.

Block length negative

The length of the block is determined by adding the quantity K from the instruction to the contents of register Bj. The addition is performed in an 18 bit ones complement mode. The resultant sum is treated as an 18 bit positive integer. A negative result will therefore appear as a large positive integer. In this case the SCM block range condition flag, and possibly the LCM block range condition flag, will set in the PSD register, indicating too large a block for the assigned fields. The block copy instruction will issue as a pass. The exchange jump to (EEA) resulting from setting the SCM block range condition flag will not occur before execution of the next program instruction word unless a delay is introduced by subsequent instructions in the current instruction word.

Block length zero

A zero block length is treated as a normal situation. No error flags are set. The block copy instruction is executed as a pass.

Last parcel

The block copy instruction requires two parcels of an instruction word for normal use. If this instruction begins in the first, second, or third parcel of an instruction word the following parcel completes the instruction. If a block copy instruction begins in the last parcel of an instruction word it will not be continued in the following word. In this case the instruction will be executed as if there were a fifth parcel in the instruction word and this parcel contained all zeros.

Error condition during execution

A LCM or SCM parity error may occur during the execution of a block copy instruction. An arithmetic error from a previous instruction may also occur during the beginning of the block copy sequence. If any error conditions occur, the proper flags are set in the PSD register, and the block copy instruction is executed to completion. There are no error conditions which will interrupt the instruction before completion.

I/O interrupt during execution

An I/O section interrupt request may occur during the execution of a block copy instruction. In this case the interrupt request is not honored until the block copy instruction has been completed and any subsequent instructions in the current instruction word have been completed.

013jx	xxxxx

Exchange exit (exit mode flag set)

This is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction causes the current program sequence to terminate with an exchange jump to an address in the SCM field for the current program. The exchange package in this case is located at relative address K + (Bj) in the SCM field. The two quantities, K and (Bj), are added in an 18 bit ones complement mode. The result is treated as an 18 bit positive integer. This integer is added to (RAS), also treated as an 18 bit positive integer, to form the absolute address of the exchange package in SCM.

This form of the 013 instruction is used by the monitor program only. The exit mode flag in the PSD register is cleared during execution of object programs. The monitor program uses this instruction to exchange jump to one of a number of possible object program exchange packages. Each of these exchange packages will normally specify a cleared exit mode flag. A selected object program exchange package will then return to this same area of SCM and resume the monitor program when its execution interval has been completed (see alternate form of 013 instruction).

This instruction has priority over all other types of exchange jump requests. If an I/O interrupt request or an error exit request has occurred prior to the execution of this instruction, this request is denied, and the exchange jump specified by this instruction is executed. The rejected interrupt request is not lost in this process since the conditions which caused it will be reinstated when the exchange package enters its next execution interval.

The remaining instructions, if any, in the current program instruction word will not be executed. The program address stored in the exchange package for the current program will be advanced one count from the address of the current instruction word. The program will therefore continue at the first parcel of the following instruction word during the next execution interval for this exchange package.

The current contents of the instruction word stack are voided by the execution of this instruction.

Execution time

The minimum execution time for this instruction is 28 clock periods. This is the minimum time from the arrival of this instruction in the upper parcel of the CIW register until the arrival of the first instruction for the next program. This instruction issues from the CIW register in the second clock period of the sequence. A 60 bit word of all zeros is read into the CIW register at this time, voiding any following instructions in the current instruction word. The IAS is cleared to all zeros, which voids the contents of the IWS. No further instructions can enter the CIW register until the exchange sequence has been completed and the IWS is loaded with a new sequence of instructions.

The following is a chronological listing of events in the execution of this instruction for the case of minimum execution time.

CP00 013 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Register Bj free.
Set PXF.
Set RIF.

CP01 013 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit a blank word to the CIW register.
Transmit K + (Bj) + (RAS) to the XJA register.
RIF remains set.
Clear PXF.

CP02 No instruction in the upper parcel of the CIW register
All operating registers free.
FIF not set.
Set XSF.
Clear RIF.

No instruction in the upper parcel of the CIW register
Void (IWS). Clear (IAS).
Transmit (XJA) to the BAK register.
Transmit (BPA), (A0), and (P) to the SWA register.
Advance (XSK) to 01.
XSF remains set.
Set XIF.

No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Transmit (B1), (A1), and (RAS) to the SWA register.
Advance (XSK) to 02.
XSF remains set.
XIF remains set.

CP05 No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
First SCM address leaves SAS for SCM bank address register.
Transmit (B2), (A2), and (FLS) to the SWA register.
Advance (XSK) to 03.
XSF remains set.
XIF remains set.

CP06 No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Begin read/write cycle for first reference in SCM bank.
Second SCM address leaves SAS for SCM bank address register.
Transmit (B3), (A3), and (PSD) to the SWA register.
Advance (XSK) to 04.
XSF remains set.
XIF remains set.

CP07 No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Begin read/write cycle for second reference in SCM bank.
Third SCM address leaves SAS for SCM bank address register.
Transmit (B4), (A4), and (RAL) to the SWA register.
Advance (XSK) to 05.
XSF remains set.
XIF remains set.

No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Begin read/write cycle for third reference in SCM bank.
Fourth SCM address leaves SAS for SCM bank address register.
Transmit (B5), (A5), and (FLL) to the SWA register.
Advance (XSK) to 06.
XSF remains set.
XIF remains set.

CP09 No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
First SCM reference reads to SCM bank operand register.
Begin read/write cycle for fourth reference in SCM bank.
Fifth SCM address leaves SAS for SCM bank address register.
Transmit (B6), (A6), and (NEA) to the SWA register.
Advance (XSK) to 07.
XSF remains set.
XIF remains set.

No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Transmit first SCM reference word to the SRO register.
Transmit first word from SWS to SCM bank operand register.
Second SCM reference reads to SCM bank operand register.
Begin read/write cycle for fifth reference in SCM bank.
Sixth SCM address leaves SAS for SCM bank address register.
Transmit (B7), (A7), and (EEA) to the SWA register.
Advance (XSK) to 10.
XSF remains set.
XIF remains set.

No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Transmit (SRO) to the BPA, AO, and P registers.
Transmit second SCM reference word to the SRO register.
Transmit second word from SWS to SCM bank operand register.
Third SCM reference reads to SCM bank operand register.
Begin read/write cycle for sixth reference in SCM bank.
Seventh SCM address leaves SAS for SCM bank address register.
Transmit (XO) to the SWA register.
Advance (XSK) to 11.
XSF remains set.
XIF remains set.

CP12 No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Transmit (SRO) to the B1, A1, and RAS registers.
Transmit third SCM reference word to the SRO register.
Transmit third word from SWS to SCM bank operand register.
Fourth SCM reference reads to SCM bank operand register.
Begin read/write cycle for seventh reference in SCM bank.
Eighth SCM address leaves SAS for SCM bank address register.
Transmit (X1) to the SWA register.
Advance (XSK) to 12.
XSF remains set.
XIF remains set.

CP13 No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Transmit (SRO) to the B2, A2, and FLS registers.
Transmit fourth SCM reference word to the SRO register.
Transmit fourth word from SWS to SCM bank operand register.
Fifth SCM reference reads to SCM bank operand register.
Begin read/write cycle for eighth reference in SCM bank.
Ninth SCM address leaves SAS for SCM bank address register.
Transmit (X2) to the SWA register.
Advance (XSK) to 13.
XSF remains set.
XIF remains set.

CP14 No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Transmit (SRO) to the B3, A3, and PSD registers.
Transmit fifth SCM reference word to the SRO register.
Transmit fifth word from SWS to SCM bank operand register.
Sixth SCM reference reads to SCM bank operand register.
Begin read/write cycle for ninth reference in SCM bank.
10th SCM address leaves SAS for SCM bank address register.
Transmit (X3) to the SWA register.
Advance (XSK) to 14.
XSF remains set.
XIF remains set.

CP15 No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Transmit (SRO) to the B4, A4, and RAL registers.
Transmit sixth SCM reference word to the SRO register.
Transmit sixth word from SWS to SCM bank operand register.
Seventh SCM reference reads to SCM bank operand register.
Begin read/write cycle for 10th reference in SCM bank.
11th SCM address leaves SAS for SCM bank address register.
Transmit (X4) to the SWA register.
Advance (XSK) to 15.
XSF remains set.
XIF remains set.

CP16 No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Transmit (SRO) to the B5, A5, and FLL registers.
Transmit seventh SCM reference word to the SRO register.
Transmit seventh word from SWS to SCM bank operand register.
Eighth SCM reference reads to SCM bank operand register.
Begin read/write cycle for 11th reference in SCM bank.
12th SCM address leaves SAS for SCM bank address register.
Transmit (X5) to the SWA register.
Advance (XSK) to 16.
XSF remains set.
XIF remains set.

CP17 No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Transmit (SRO) to the B6, A6, and NEA registers.
Transmit eighth SCM reference word to the SRO register.
Transmit eighth word from SWS to SCM bank operand register.
Ninth SCM reference reads to SCM bank operand register.
Begin read/write cycle for 12th reference in SCM bank.
13th SCM address leaves SAS for SCM bank address register.
Transmit (X6) to the SWA register.
Advance (XSK) to 17.
XSF remains set.
XIF remains set.

CP18 No instruction in the upper parcel of the CIW register.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Transmit (SRO) to the B7, A7, and EEA registers.
Transmit ninth SCM reference word to the XO register.
Transmit ninth word from SWS to SCM bank operand register.
10th SCM reference reads to SCM bank operand register.
Begin read/write cycle for 13th reference in SCM bank.
14th SCM address leaves SAS for SCM bank address register.
Transmit (X7) to the SWA register.
Clear (XSK).
Set JCF.
Clear XSF.
XIF remains set.

No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
Set OSF.
Set JOF.
Transmit (P) + (RAS) to the IFA register.
Set F1F, F2F.
Clear JCF.
Transmit (BAK) to the SAS. Tag for exchange jump.
Advance (BAK).
Transmit 10th SCM reference word to the X1 register.
Transmit 10th word from SWS to SCM bank operand register.
11th SCM reference reads to SCM bank operand register.
Begin read/write cycle for 14th reference in SCM bank.
15th SCM address leaves SAS for SCM bank address register.
Clear XIF.

No instruction in the upper parcel of the CIW register.
Transmit (P) to the NSA register.
Set M1F, M2F.
Clear JOF.
The OSF remains set.
Transmit (IFA) to the SAS. Tag for read to IWS.
Advance (IFA).
Clear F2F.
Transmit 11th SCM reference word to the X2 register.
Transmit 11th word from the SWS to SCM bank operand register.
12th SCM reference reads to SCM bank operand register.
Begin read/write cycle for 15th reference in SCM bank.
16th SCM address leaves SAS for SCM bank address register.

CP21 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Transmit (IFA) to the SAS. Tag for read to IWS.
Advance (IFA).
First fetch address leaves the SAS for a SCM bank.
Clear F1F.
Transmit 12th SCM reference word to the X3 register.
Transmit 12th word from the SWS to SCM bank operand register.
13th SCM reference reads to SCM bank operand register.
Begin read/write cycle for 16th reference in SCM bank.

CP22 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
First fetch memory read/write cycle begins in a SCM bank.
Second fetch address leaves the SAS for a SCM bank.
Transmit 13th SCM reference word to the X4 register.
Transmit 13th word from the SWS to SCM bank operand register.
14th SCM reference reads to SCM bank operand register.

No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Second fetch memory read/write cycle begins in a SCM bank.
Transmit 14th SCM reference word to the X5 register.
Transmit 14th word from the SWS to SCM bank operand register.
15th SCM reference reads to SCM bank operand register.

CP24 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Transmit 15th SCM reference word to the X6 register.
Transmit 15th word from the SWS to SCM bank operand register.
16th SCM reference reads to SCM bank operand register.

CP25 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
First fetch instruction word reads to SCM bank operand register.
Transmit 16th SCM reference word to the X7 register.
Transmit 16th word from the SWS to SCM bank operand register.

CP26 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Transmit the first fetch instruction word to the IWS.
Transmit (NSA) to the IAS.
Shift the IWS and the IAS one word position.
Advance (NSA).
Second fetch instruction word reads to SCM bank operand register.
Clear M2F.

CP27 No instruction in the upper parcel of the CIW register.
Coincidence in the IAS.
Read a valid instruction word from the IWS to the CIW register.
Clear the OSF.
Transmit the second fetch instruction word to the IWS.
Transmit (NSA) to the IAS.
Shift the IWS and the IAS one word position.
Advance (NSA).
Clear M1F.

CP28 Next instruction in the upper parcel of the CIW register.
Instruction may issue.

Execution delays

The PXF will not set in this mode of the 013 instruction until the Bj register is free. The RIF will set without this condition, but the sequence will not proceed to the commands listed for CP01 until the PXF is set.

The XSF will not set as indicated in CP02 in the above sequence until two conditions are satisfied:

All operating registers must be free: This implies that all previously issued instructions have delivered their results to the operating registers. This will normally be the case in the clock period CP02 because of the delay from the preceding two clock periods.

The F1F must not be set: This will be the case unless a SCM bank conflict has caused a SAS backup condition which prevented an instruction fetch address from leaving the IFA register. This condition is not likely to occur often enough to cause a significant average delay.

A delay will occur during the flow of data in the exchange sequence if a SCM bank conflict exists. This will occur if a SCM bank is still completing a read/write cycle from a previous reference at the time the exchange sequence reference is made to that bank. The remainder of the exchange sequence is delayed in this case until the conflict is resolved.

The exchange sequence has priority over I/O section word requests for SCM. If an I/O section word request occurs during the block of 16 exchange sequence entries to the SAS, this request is not honored until the 16th address has arrived at the SAS.

There may be a delay in reading the instruction fetches from SCM because of a SCM bank conflict. A conflict may occur between the instruction fetch addresses and the exchange sequence addresses. An I/O section word request has priority over instruction fetch references to SCM. An I/O section reference to SCM may therefore occur between the last word reference in the exchange sequence and the first reference for instruction fetch. In this case a one clock period delay results from the address insertion to the SAS and a possible nine clock period delay if the I/O section reference and the instruction fetch reference are to the same bank.

Special situations

Exchange address out of range

There is no protection for addressing out of the SCM field on this instruction. Any error in the calculation of the exchange package address, either in range, or out of range, will almost certainly result in complete system failure. The exchange package address is determined by adding K to (Bj) in a ones complement mode. The result is treated as an 18 bit positive integer and is added to (RAS), also treated as an 18 bit positive integer. The lowest order 16 bits of this last addition are used as the absolute address in SCM for the exchange package.

Last parcel

This instruction normally requires two parcels of an instruction word. If this instruction begins in the first, second, or third parcel of an instruction word the following parcel completes the instruction. If this instruction begins in the last parcel of an instruction word it will not be continued in the following word. In this case the instruction will be executed as if there were a fifth parcel in the instruction word and this parcel contained all zeros.

Error condition

This instruction takes priority over an error exit request. The flag, or flags, associated with the error exit request are preserved in the exchange package. The error exit request will be regenerated at the beginning of the next execution interval for the exchange package. This will cause an error exit in the next execution interval for the exchange package before the execution of the first instruction.

I/O section interrupt

This instruction takes priority over an I/O section interrupt request. The I/O section request is not honored until the exchange jump has been completed and a new exchange package has been loaded into the computation section of the CPU.

01300 Exchange exit (exit mode flag cleared)

An exchange exit instruction executed in this mode causes the current program sequence to terminate with an exchange jump to address (NEA). This is an absolute address in SCM and is generally not in the SCM field for the current program. This mode makes no use of the j or k designators in the instruction.

This instruction is the vehicle for switching rapidly from an object program to a monitor program. All operating register values, program address, and mode selections are preserved in this process in order that the object program may be continued at a later time. The program address in the object program exchange package will be advanced one count from the address of the instruction word containing the exchange exit instruction. The monitor program will normally resume the object program at this address.

This instruction is intended for use in calling the system monitor program for input-output requests, library calls, storage assignments, etc. The operating register values at the time of execution of this instruction are intended as the vehicle for parameter interchange between the object program and the monitor program.

This instruction has priority over all other types of exchange jump requests. If an I/O interrupt request or an error exit request has occurred prior to the execution of this instruction, this request is denied and the exchange jump specified by this instruction is executed. The rejected interrupt request is not lost in this process since the conditions which caused it will be reinstated when the exchange package enters its next execution interval.

The remaining instructions, if any, in the current instruction word will not be executed. The program address stored in the exchange package for the current program will be advanced one count from the address of the current instruction word. The program will therefore continue at the first parcel of the following instruction word during the next execution interval for this exchange package unless the monitor program alters the exchange package.

The current contents of the instruction word stack are voided by the execution of this instruction.

Execution time

The minimum execution time for this instruction is 28 clock periods. This is the minimum time from the arrival of this instruction in the upper parcel of the CIW register until the arrival of the first instruction for the next program. This instruction issues from the CIW register in the second clock period of the sequence. A 60 bit word of all zeros is read into the CIW register at this time, voiding any following instructions in the current instruction word. The IAS is cleared to all zeros, which voids the contents of the IWS. No further instructions can enter the CIW register until the exchange sequence has been completed and the IWS is loaded with a new sequence of instructions.

Timing considerations for this mode of the exchange exit instruction are essentially the same as for the alternate mode. The only differences in the command timing occur in the initial clock periods of the sequence. These are listed below for the case of minimum execution time.

```
CP00  013 instruction in the upper parcel of the CIW register.
      Instruction does not issue.
      Set PXF.
      Set RIF.

      013 instruction in the upper parcel of the CIW register.
      Instruction issues.
      Transmit a blank word to the CIW register.
      Transmit (NEA) to the XJA register.
      RIF remains set.
      Clear PXF.

      No instruction in the upper parcel of the CIW register.
      All operating registers free.
      All instruction fetches completed.
      Set XSF.
      Clear RIF.
```

(See alternate mode of 013 instruction for rest of listing)

Execution delays

The XSF will not set as indicated in CP02 in the above sequence until two conditions are satisfied:

All operating registers must be free: This implies that all previously issued instructions have delivered their results to the operating registers. This will normally be the case in the clock period CP02 because of the delay from the preceding two clock periods.

The F1F must not be set: This will be the case unless a SCM bank conflict has caused a SAS backup condition which prevented an instruction fetch address from leaving the IFA register. This condition is not likely to occur often enough to cause a significant average delay.

A delay will occur during the flow of data in the exchange sequence if a SCM bank conflict exists. This will occur if a SCM bank is still completing a read/write cycle from a previous reference at the time the exchange sequence reference is made to that bank. The remainder of the exchange sequence is delayed in this case until the conflict is resolved.

The exchange sequence has priority over I/O section word requests for SCM. If an I/O section word request occurs during the block of 16 exchange sequence entries to the SAS, this request is not honored until the 16th address has arrived at the SAS.

There may be a delay in reading the instruction fetches from SCM because of a SCM bank conflict. A conflict may occur between the instruction fetch addresses and the exchange sequence addresses. An I/O section word request has priority over instruction fetch references to SCM. An I/O section reference to SCM may therefore occur between the last word reference in the exchange sequence and the first reference for instruction fetch. In this case a one clock period delay results from the address insertion to the SAS and a possible nine clock period delay if the I/O section reference and the instruction fetch reference are to the same bank.

Special situations

Designator j, k, not zero

A nonzero j or k designator will have no effect on the results of this instruction. If the j designator is nonzero a test will be made for register Bj free. This may delay execution of the instruction but will not affect the results.

(NEA) out of range

There are no protective tests made on the exchange jump address for this instruction. The assignment of (NEA) is a responsibility of the system monitor program. Normally the SCM field for an object program does not include the address (NEA). If (NEA) has more than 16 bits of significance, considered as a positive integer, the upper bits are discarded and the lower 16 bits used as the absolute address in SCM for the exchange jump.

Error condition

This instruction takes priority over an error exit request. The flag, or flags, associated with the error exit request are preserved in the exchange package. The error exit request will be regenerated at the beginning of the next execution interval for the exchange package. This will cause an error exit in the next execution interval for the exchange package before the execution of the first instruction.

I/O section interrupt

This instruction takes priority over an I/O section interrupt request. The I/O section request is not honored until the exchange jump has been completed and a new exchange package has been loaded into the computation section of the CPU. The I/O section request will then be honored before execution of the first instruction in the new program.

014jk Read LCM

This instruction reads one word from the LCM and enters this word in an X register. The word is read from the LCM field at relative address (Xk). The word is then entered in register Xj. The SCM is not involved in this process.

This instruction is intended for direct addressing of the LCM for individual words. It may also be used to advantage in addressing a string of words in consecutive storage locations. This is particularly true if a string of words is to be read, modified, and written back into the same storage locations. The process of reading and writing will proceed in this case without a LCM read/write cycle delay until the addressing crosses a LCM bank boundary.

This instruction is buffered to the extent that it issues in one clock period unless a previous LCM reference is in process. When this instruction issues the LCM busy flag is set and remains set until the requested word has been delivered to the designated X register. The destination X register is reserved in a manner analogous to a reference to the SCM. This process differs from a SCM read reference in that only one LCM read or write may be in process at one time.

Execution time

A read LCM instruction for a word not currently residing in one of the LCM bank operand registers will normally require 14 clock periods to deliver the requested word to the designated X register. A LCM bank read/write cycle will be initiated in the proper LCM bank, and a field of 8 words will be read to the corresponding LCM bank operand register. A read LCM instruction for a word already residing in a LCM bank operand register as a result of a previous instruction will require 3 clock periods to deliver the requested word to the designated X register.

The following is a chronological listing of events in a read LCM instruction in which the required word is already residing in a LCM bank operand register as a result of a previous instruction.

```
CP00  014 instruction in the upper parcel of the CIW register.
      LCM busy flag not set.
      Xj register free.
      Xk register free.
      Instruction issues.
      Transmit next instruction to upper parcel of CIW register.
      Transmit (Xk) + (RAL) to LCM word address register.
      Set LCM busy flag.
      Set Xj reservation flag.

      Next instruction in the upper parcel of the CIW register.
      Instruction may issue.
      Test LCM word address against LCM bank addresses.
      Transmit bank address to LCM bank address register.
      LCM busy flag remains set.

      Transmit word from LCM bank operand register to Xd register.
      Clear LCM busy flag.
      Clear Xd reservation flag.
```

The following is a chronological listing of events in a read LCM instruction in which the required word is not residing in a LCM bank operand register.

```
CP00  014 instruction in the upper parcel of the CIW register.
      LCM busy flag not set.
      Xj register free.
      Xk register free.
      Instruction issues.
      Transmit next instruction to upper parcel of CIW register.
      Transmit (Xk) + (RAL) to LCM word address register.
      Set LCM busy flag.
      Set Xj reservation flag.

CP01  Next instruction in the upper parcel of the CIW register.
      Instruction may issue.
      LCM busy flag remains set.

      Transmit bank address to LCM bank address register.

      Begin LCM bank read/write cycle.
      LCM busy flag remains set.

      LCM bank reads 8 words to LCM bank operand register.
      LCM busy flag remains set.
```

CP14 Requested word leaves LCM bank operand register for Xd.
LCM busy flag remains set.

CP15 Requested word arrives at Xd register
Clear LCM busy flag.
Clear Xd reservation flag.

CP65 Complete LCM bank read/write cycle.

Execution delays

This instruction will not issue, and the sequences listed will not begin, until three conditions have been satisfied:

LCM busy flag not set: This will be true unless another LCM (011 012, 014, 015) instruction has just preceded this instruction.

Xj register free: This will normally be true since the result is to be stored here.

Xk register free: This will depend on the previous instructions and will involve a one clock period delay if the preceding instruction was an increment instruction to this destination register.

A delay will occur at CP01 in the second sequence listed above if the LCM bank required for the requested word is still completing a read/write cycle from a previous reference. The bank address will be transmitted to the LCM bank address register as soon as the bank read/write cycle has been completed.

A delay will occur in either of the two sequences listed above if another functional unit is transmitting data to an X register in the same clock period in which the LCM transmission is indicated. In this case the LCM transmission is blocked and is repeated each clock period until the X register input path is free.

Special situations

(Xk) negative
(Xk) greater than 19 significant bits

The lowest order 19 bits of (Xk) are used to determine the address in the LCM field. The higher order bits are ignored. If (Xk) is negative the lowest order 19 bits are masked out and treated as a positive integer. No error flags are set for these conditions unless the resulting address is out of range.

Address out of range

The lowest order 19 bits of (Xk) are compared with (FLL) to determine if the requested address is in the assigned LCM field. If the requested address is greater than, or equal to, (FLL) the LCM direct range condition flag is set in the PSD register. This flag will cause an error exit request to interrupt the program with an exchange jump to address (EEA). The instruction will be executed in this case with a LCM read reference beyond the assigned field, and a word will be entered in the Xj register from this location. The absolute address in LCM for this reference will be the lowest order 19 bits in the sum resulting from adding (RAL) to the lowest order 19 bits of (Xk). The exchange jump resulting from the error exit request will generally not occur before one or more subsequent instructions have been executed.

Use of the X0 register

The X0 register may be used for either Xj or Xk in this instruction.

Xj and Xk same register

The j and k designators may have the same value in this instruction. In this case the requested address is lost when the word arrives at the Xj register.

Read from block copy field

The requested word may reside in a LCM bank operand register as a result of a previous block copy instruction. This condition is sensed, and the word is read directly from the LCM bank operand register in this case.

015jk Write LCM

This instruction writes one word directly into LCM from an X register. The word is read from register Xj and is written into the LCM field at relative address (Xk). The SCM is not involved in this process.

This instruction is intended for direct addressing of the LCM for individual words. It may also be used to advantage in addressing a string of words in consecutive storage locations. This is particularly true if a string of words is to be read, modified, and written back into the same storage locations. The process of reading and writing will proceed in this case without a LCM bank read/write cycle delay until the addressing crosses a LCM bank boundary.

This instruction is buffered to the extent that it issues in one clock period unless a previous LCM reference is in process. When this instruction issues the LCM busy flag is set and remains set until the word has been delivered to the proper LCM bank operand register. No X register reservations are made for this instruction. The following instruction may issue in the next clock period and may use either of the X registers designated in this instruction. If the word cannot be entered immediately in the proper LCM bank operand register it is held in the LCM write register until the LCM bank operand register is free. This process differs from a SCM write reference in that only one LCM read or write may be in process at one time.

Execution time

This instruction normally requires 3 clock periods to deliver the word to the proper LCM bank operand register. The instruction normally issues in one clock period and sets the LCM busy flag. The LCM busy flag remains set for two clock periods. A subsequent LCM instruction may then begin 3 clock periods after this instruction. A delay in clearing the LCM busy flag will occur if the required LCM bank is busy completing a bank read/write cycle for a different block of 8 words than that required for this instruction. The LCM busy flag will clear as soon as the LCM bank is free.

The following is a chronological listing of events in a write LCM instruction for the case of minimum execution time.

CP00 015 instruction in the upper parcel of the CIW register.
LCM busy flag not set.
Xj register free.
Xk register free.
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Xk) + (RAL) to LCM word address register.
Transmit (Xj) to LCM write register.
Set LCM busy flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Test LCM word address against LCM bank addresses.
Transmit bank address to LCM bank address register.
Word leaves LCM write register for bank operand register.
LCM busy flag remains set.

CP02 Begin read/write cycle in LCM bank.
Word arrives at LCM bank operand register.
Clear LCM busy flag.

Execution delays

This instruction will not issue, and the sequence listed will not begin, until three conditions have been satisfied:

LCM busy flag not set: This will be true unless another LCM (011, 012, 014, 015) instruction has just preceded this instruction.

Xj register free: This will depend on previous instruction timing and may vary from no delay to a 19 clock period delay for a just issued divide result.

Xk register free: This will depend on the previous instructions and will involve a one clock period delay if the preceding instruction was an increment instruction to this destination register.

A delay will occur at CP01 in the sequence listed above if the required LCM bank is busy completing a bank read/write cycle for a different block of 8 words than that required for this instruction. In this case the word will be held in the LCM write register until the LCM bank is free.

Special situations

(Xk) negative
(Xk) greater than 19 significant bits

The lowest order 19 bits of (Xk) are used to determine the address in the LCM field. The higher order bits are ignored. If (Xk) is negative the lowest order 19 bits are masked out and treated as a positive integer. No error flags are set for these conditions unless the resulting address is out of range.

Address out of range

The lowest order 19 bits of (Xk) are compared with (FLL) to determine if the requested address is in the assigned LCM field. If the requested address is greater than, or equal to, (FLL) the LCM direct range condition flag is set in the PSD register. This flag will cause an error exit request to interrupt the program with an exchange jump to address (EEA). In this case the word will not be written into LCM. The exchange jump resulting from the error exit condition will generally not occur before one or more subsequent instructions have been executed.

Use of the XO register

The XO register may be used for either Xj or Xk in this instruction.

Xj and Xk same register

The j and k designators may have the same value in this instruction. In this case the requested address is also the operand.

0160k Reset input buffer

This instruction resets the channel (Bk) input buffer in preparation for the next incoming record. The channel (Bk) input buffer address register is cleared to zero. The channel input assembly register is reset to first position.

This instruction is intended for execution in the monitor program input routine which terminates a record of incoming data and prepares for the next record. The monitor input routine is called by an I/O section interrupt request when the record flag is set on the channel input data path. The data in the channel input buffer is then normally transferred to the LCM, and this instruction is executed to clear the buffer for the next incoming record.

This instruction is effective only if the monitor mode flag is set in the PSD register. If the monitor mode flag is cleared this instruction becomes a pass instruction. There are no interlocks for this instruction other than the monitor mode flag. When this instruction issues it will execute the required channel functions without regard to the current status or activity at the channel input register.

This instruction is normally never executed except in response to an I/O section interrupt request resulting from the setting of the channel input record flag. The record flag is cleared when the interrupt request is generated. Further entries to the channel input buffer are not locked out by the interrupt request flag in the channel access control during the execution interval for the interrupt exchange package. The PPU must wait for a positive response from the monitor program over the output channel before beginning the next record.

Execution time

This instruction requires four clock periods to clear the channel input buffer address register and reset the channel input assembly register. The timing for the events in the execution of this instruction is as follows.

CP00 0160 instruction in the upper parcel of the CIW register.
Monitor mode flag set.
Bk register free.
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Bk) to I/O section access control.
Set go reset channel flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Clear go reset channel flag.

Go reset signal arrives at channel access control.

CP03 Clear channel input buffer address register.
Reset channel input assembly register.

Execution delays

This instruction will not issue, and the sequence will not begin, until the Bk register is free. This register contains the channel number and will generally be a constant during the execution interval for this monitor exchange package. A delay in this instruction issue is unlikely.

Special situations

(Bk) not a valid channel number

The lowest order four bits of (Bk) are used in this instruction. The higher order bits are ignored. If higher order bits are set in (Bk) the lowest order four bits are masked out and used to determine the channel number. If (Bk) = 0, this instruction becomes a pass instruction.

Monitor mode flag not set

If the monitor mode flag is not set in the PSD register when this instruction is executed, this instruction becomes a pass instruction.

Channel active

The channel input buffer is normally inactive when this instruction is executed because the PPU has transmitted a record flag and is waiting for monitor response on the output channel. If the PPU has for some reason continued transmitting data, a word may be waiting to enter the channel input buffer and a word request flag may be set. These two operations may occur in the same clock period with conflicting commands to the registers from the channel access control. In this case the commands associated with this instruction take priority, and the result is a loss of data in the input buffer for the incoming record. The incoming record will continue in this case with no indication of error except that the record will be shortened by the lost data.

Consecutive resets for different channels

Two or more reset input buffer instructions may occur in consecutive program instruction locations referencing different channels. These instructions may issue in consecutive clock periods, and no interference will result in the I/O section access control.

Consecutive resets for same channel

Two or more reset input buffer instructions may occur in consecutive program instruction locations referencing the same channel. These instructions will issue in consecutive clock periods and repeatedly perform the same functions. No interference will occur other than the obvious repetitive functions.

016jk Read channel input status (j nonzero)

This instruction reads the current value of the channel (Bk) input buffer address register contents to register Bj. The status of the channel (Bk) input buffer address register is not altered.

This instruction is intended for use in monitoring the progress of the channel input buffer. The channel input buffer area is divided into two fields by the threshold testing mechanism. The first half of the buffer area constitutes one field and the last half of the buffer area the other field. An I/O section interrupt request is generated by the threshold testing mechanism whenever the channel input buffer address is advanced across a field boundary. This will occur at the center of the buffer area and at the end of the buffer area.

This instruction is the only vehicle for a monitor program to determine whether an I/O section interrupt request was generated by a buffer threshold test or by a record flag. The monitor program must retain the buffer address from one interrupt period to the next. If the buffer address is in the same field as for the previous interrupt, the interrupt request was from a record flag. If the buffer address is in the opposite field from the previous interrupt, the interrupt request was from a threshold test.

This instruction has a special use if the channel number (Bk) is zero. There are no buffer areas for the MCU which use the I/O section channel zero access position. In this case the current contents of the CPU clock period counter are read into the Bj register. This is a 17 bit counter which is advanced one count in a twos complement mode each clock period. This count is intended for timing measurements in CPU programs. Timing considerations for this special use are the same as the normal timing from a channel input buffer address register.

Execution time

This instruction requires 3 clock periods to deliver the channel input buffer address to the Bj register. The timing for the events in the execution of this instruction is as follows.

```
CP00  016 instruction in the upper parcel of the CIW register.
      Bk register free.
      Bj register free.
      Instruction issues.
      Transmit next instruction to upper parcel of CIW register.
      Transmit (Bk) to I/O section access control.
      Set go read channel flag.
      Set Bj reservation flag.

      Next instruction in the upper parcel of the CIW register.
      Instruction may issue.
      Clear go read channel flag.

      Channel input buffer address arrives at Bd register.
      Clear Bd reservation flag.
```

Execution delays

This instruction will not issue, and the sequence will not begin, until the Bj and Bk registers are free. These registers will normally both be free since one is the result destination and the other the channel number.

Special situations

(Bk) not a valid channel number

The lowest order four bits of (Bk) are used in this instruction. The higher order bits are ignored. If higher order bits are set in (Bk) the lowest order four bits are masked out and used to determine the channel number. If (Bk) = 0, this instruction reads the contents of the CPU clock period counter.

Consecutive executions

Two or more read channel input status instructions may occur in consecutive program instruction locations referencing the same or different channels. These instructions may issue in consecutive clock periods providing the Bj register reservations do not cause a delay. No interference will result in the I/O section access control in these situations.

0170k Reset output buffer

This instruction resets the channel (Bk) output buffer in preparation for the next record transmission. The channel (Bk) output buffer address register is cleared to zero. A record pulse is transmitted over the channel output data path. The channel output word request flag is then set to read the first word from the channel output buffer.

This instruction is intended for execution in the monitor program output routine to initiate a new record transmission over a channel output data path. The channel output buffer is normally inactive when this instruction is executed. The channel output buffer is loaded with the data for the next record, and this instruction is executed to initiate the transmission. A record pulse is transmitted at the time this instruction is executed to indicate the beginning of a new record. The first word of data will follow as soon as the channel output word request flag has caused the first word to be read from the output buffer to the channel output disassembly register.

This instruction is effective only if the monitor mode flag is set in the PSD register. If the monitor mode flag is cleared this instruction becomes a pass instruction. There are no interlocks for this instruction other than the monitor mode flag. When this instruction issues it will execute the required channel functions without regard to the current status or activity at the channel output register. The channel output disassembly register is reset by the channel output word request flag.

Execution time

This instruction requires four clock periods to clear the channel output buffer address register. A record pulse is transmitted over the channel output data path in the fourth clock period. If no conflicts occur the first word pulse is transmitted over the channel output data path in the 16th clock period. The timing for the events in the execution of this instruction is as follows.

CP00 0170 instruction in the upper parcel of the CIW register.
Monitor mode flag set.
Bk register free.
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Bk) to I/O section access control.
Set go reset channel flag.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Clear go reset channel flag.

Go reset signal arrives at channel access control

CP03 Clear channel output buffer address register.
Transmit record pulse over output data path.
Set channel output word request flag.

Reset channel output disassembly register.

Set IOF.

Transmit first buffer address to the SAS. Tag for output.
Clear IOF.
Set word accepted flag in I/O access control.

Address leaves the SAS for a SCM bank address register.

CP08 Begin read/write cycle for first word in SCM bank.

First word reads from SCM bank to SCM bank operand register.

First word leaves SCM bank operand register for SRO register.

CP13 Transmit (SRO) to I/O output buffer register.
Clear channel output word request flag.

CP14 Transmit first word to channel output disassembly register.
Clear word accepted flag in I/O access control.

Transmit word pulse over output data path.

Execution delays

This instruction will not issue, and the sequence will not begin, until the Bk register is free. This register contains the channel number and will generally be a constant during the execution interval for this monitor exchange package.

The IOF will not set for this channel in CP05 in the sequence listed above if a higher priority channel also has a word request flag set. Each channel word request requires a minimum of two clock periods to clear through the I/O access control and release this mechanism for the next request.

The first buffer address will be delayed in entering the SAS in CP06 in the sequence listed above if a higher priority address is on the way to the SAS in this clock period. The addresses with higher priority are from the increment unit, return jump exit address, and exchange sequence address.

A delay will occur in entering the SAS if a backup condition exists in CP06 in the sequence above. A delay will also occur in CP07 if this address causes a storage bank conflict in SCM.

Special situations

(Bk) not a valid channel number

The lowest order four bits of (Bk) are used in this instruction. The higher order bits are ignored. If higher order bits are set in (Bk) the lowest order four bits are masked out and used to determine the channel number. If (Bk) = 0, this instruction becomes a pass instruction.

Monitor mode flag not set

If the monitor mode flag is not set in the PSD register when this instruction is executed, this instruction becomes a pass instruction.

Channel active

The channel output buffer is normally inactive when this instruction is executed because the monitor program has detected completion of the previous record before beginning this routine. There are two methods that the monitor program can use to detect end of record. One method is to read the channel output buffer address and compare with a known record length. The other is a positive response from the peripheral unit over the corresponding channel input data path. If for some reason the channel output buffer is actively moving data over the channel output data path at the time this instruction is executed, conflicting commands may be sent to the channel registers. In this case the commands associated with this instruction have priority, and the result is a loss of data in the previous record.

Consecutive resets for different channels

Two or more reset output buffer instructions may occur in consecutive program instruction locations referencing different channels. These instructions may issue in consecutive clock periods and no interference will result in the I/O section access control.

Consecutive resets for same channel

Two or more reset output buffer instructions may occur in consecutive program instruction locations referencing the same channel. These instructions will issue in consecutive clock periods and repeatedly perform the same functions. A record pulse will be transmitted over the channel output data path for each instruction execution. The channel output buffer will be repeatedly restarted, and a data word may, or may not, be transmitted over the channel output data path depending on the timing of the instructions and the conflicts that occur.

017jk Read channel output status (j nonzero)

This instruction reads the current value of the channel (Bk) output buffer address register contents to register Bj. The status of the channel (Bk) output buffer address register is not altered.

This instruction is intended for use in monitoring the progress of the channel output buffer. The channel output buffer area is divided into two fields by the threshold testing mechanism. The first half of the buffer area constitutes one field and the last half of the buffer area the other field. An I/O section interrupt request is generated by the threshold testing mechanism whenever the channel output buffer address is advanced across a field boundary. This will occur at the center of the buffer area and at the end of the buffer area.

Execution time

This instruction requires three clock periods to deliver the channel output buffer address to the Bj register. The timing of the events in the execution of this instruction is as follows.

017 instruction in the upper parcel of the CIW register.
Bk register free.
Bj register free.
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Transmit (Bk) to I/O section access control.
Set go read channel flag.
Set Bj reservation flag.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Clear go read channel flag.

Channel output buffer address arrives at Bd register.
Clear Bd reservation flag.

Execution delays

This instruction will not issue, and the sequence will not begin, until the Bj and Bk registers are free. These registers will normally be free since one is the result destination and the other the channel number.

Special situations

(Bk) not a valid channel number

The lowest order four bits of (Bk) are used in this instruction. The higher order bits are ignored. If higher order bits are set in (Bk) the lowest order four bits are masked out and used to determine the channel number. If (Bk) = 0, this instruction reads all zeros into Bj.

Consecutive executions

Two or more read channel output status instructions may occur in consecutive program instruction locations referencing the same or different channels. These instructions may issue in consecutive clock periods providing the Bj register reservations do not cause a delay. No interference will result in the I/O section access control in these situations.

02i0x	xxxxx

Jump to B + K

This instruction is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction causes the current program sequence to terminate with a jump to address (Bi) + K in the SCM field.

This instruction is intended as a vehicle to allow computed branch point destinations. This is the only CPU instruction in which a computed parameter can specify a program branch destination address. All other jump instructions have preassigned destination addresses. Program modification to implement changes in a branch point destination address is not recommended in general because of the complications associated with the instruction stack.

The quantities (Bi) and K are added in an 18 bit ones complement mode. The result is treated as an 18 bit positive integer. This resulting sum specifies the beginning address in the SCM field for the new program sequence. The remaining instructions, if any, in the current program instruction word will not be executed. The instruction word stack is not altered by the execution of this instruction.

Execution time

One of two possible sequences will be executed for this instruction depending on whether the branch point destination address is, or is not, currently in the instruction stack. If the branch point destination address is currently in the instruction stack, this instruction may be executed in a minimum of three clock periods. If the branch point destination address is not currently in the instruction stack, and is not in the process of arriving there, the minimum execution time is 11 clock periods. If the branch point address is not in the instruction stack, but is in process to the instruction stack as a fetch instruction address, the second sequence is executed and the execution time will be between three and 11 clock periods.

The following is a chronological listing of the events in the execution of this instruction for the case of branching within the instruction stack.

CP00 02 instruction in the upper parcel of the CIW register.
Bi register free.
Instruction does not issue.
Set GJF.

02 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit (Bi) + K to the P register.
Set JCF.
Clear GJF.

02 instruction in the upper parcel of the CIW register.
Instruction issues.
Coincidence in the IAS.
Read a valid instruction word from IWS to the CIW register.
Clear JCF.

CP03 Next instruction in the upper parcel of the CIW register.
Instruction may issue.

The following is a chronological listing of the events in the execution of this instruction for the case of branching out of the instruction stack.

02 instruction in the upper parcel of the CIW register.
Bi register free.
Instruction does not issue.
Set GJF.

02 instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit (Bi) + K to the P register.
Set JCF.
Clear GJF.

02 instruction in the upper parcel of the CIW register.
Instruction issues.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
Set OSF.
Set JOF.
Transmit (P) + (RAS) to the IFA register.
Set F1F, F2F.
Clear JCF.

CP03 No instruction in the upper parcel of the CIW register.
Transmit (P) to the NSA register.
Set M1F, M2F.
The OSF remains set.
Transmit (IFA) to the SAS. Tag for read to IWS.
Advance (IFA).
Clear J0F.
Clear F2F.

No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Transmit (IFA) to the SAS. Tag for read to IWS.
Advance (IFA).
First fetch address leaves the SAS for a SCM bank.
Clear F1F.

No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
First fetch read/write cycle begins in a SCM bank.
Second fetch address leaves the SAS for a SCM bank.

CP06 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Second fetch read/write cycle begins in a SCM bank.

CP07 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.

CP08 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
First fetch instruction word reads to SCM bank operand register.

No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Transmit the first fetch instruction word to the IWS.
Transmit (NSA) to the IAS.
Shift the IWS and the IAS one word position.
Advance (NSA).
Second fetch instruction word reads to SCM bank operand register.
Clear M2F.

CP10 No instruction in the upper parcel of the CIW register.
Coincidence in the IAS.
Read a valid instruction word from the IWS to CIW register.
Clear the OSF.
Transmit the second fetch instruction word to the IWS.
Transmit (NSA) to the IAS.
Shift the IWS and the IAS one word position.
Advance (NSA).
Clear M1F.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.

Execution delays

This instruction sequence will not begin until the Bi register is free. This is true of either of the two possible sequences.

The second sequence (for branching out of the instruction stack) will be delayed at CP02 if the F1F is set during this clock period. The JOF will not set in CP02 unless the F1F is cleared. The F1F will be cleared at this time unless a SCM bank conflict has caused a SAS backup condition which delayed an instruction fetch address in leaving the IFA register. If the F1F is set in CP02, the rest of the sequence will be delayed and the JOF will not set until all fetch instruction words have arrived at the IWS and the M1F has been cleared.

The transmission of (P) to the NSA register, the setting of M1F and M2F, and the clearing of JOF will not take place in CP03 if an instruction fetch is still in process and M1F is set for this earlier reference. This will not delay the execution of the remainder of this instruction. These commands will be issued and the functions performed when the previous fetch instruction word arrives at the IWS.

The transmission of (IFA) to the SAS will be delayed in CP03 or in CP04 if a backup situation exists in the SAS. The remainder of the sequence will be delayed by the amount of time required to resolve the SCM bank conflict and clear up the SAS backlog.

A delay in the arrival of the first fetch instruction word at the IWS will occur if a SCM bank conflict prevents the first fetch address from leaving the SAS in CP04. This will delay the completion of this instruction by the time required to resolve the SCM bank conflict.

Special situations

Designator j not zero

The j designator in this instruction is normally zero. This designator is ignored, however, in the execution of the instruction, and a nonzero value will have no effect on the results.

Last parcel

This instruction requires two parcels of an instruction word for normal use. If this instruction begins in the first, second, or third parcel of an instruction word the following parcel completes the instruction. If this instruction begins in the last parcel of an instruction word it will not be continued in the following word. In this case the instruction will be executed as if there were a fifth parcel in the instruction word and this parcel contained all zeros.

I/O interrupt or error condition

If an I/O interrupt request or an error exit request exists at the time this instruction is executed, the instruction is executed to completion before the interrupt occurs.

Previous fetch is destination address

If the branch point destination address is not in the instruction stack at the beginning of this instruction sequence, but is in process to the instruction stack as an instruction fetch address, this fetch instruction word will arrive at the IWS and enter the CIW register in less than the minimum 11 clock periods normally required for a jump out of stack. The remainder of the instruction sequence will be completed and a duplicate word will be read to the IWS as a normal initial instruction fetch. This will not cause any special problems in the IWS.

Jump out of range

If the branch point destination address is greater than the SCM field length the program range condition flag is set in the PSD register. The instruction will execute to completion, but the first instruction word for the next program sequence will not read from the IWS to the CIW register. At this point an error interrupt will occur as a result of the program range condition flag, and an exchange jump will occur to address (EEA) in the SCM. The terminating exchange package will contain the out-of-range address in the program address field.

Jump to zero

A jump to relative address zero in the SCM field is treated in the same manner as a jump out of range. The program range condition flag is set in the PSD register, and the program will be terminated with an error exit to address (EEA). The terminating exchange package will contain a zero quantity in the program address field.

Jump to breakpoint address

A jump to address (BPA) will set the breakpoint condition flag in the PSD register. The instruction will be executed to completion, and the exchange jump to address (EEA) will occur before the first instruction is executed at the branch point destination address.

030jx	xxxxx

Branch on X zero

This instruction is a two parcel instruction in which the lower order 18 bits are used as an operand K. Execution of this instruction will cause the program sequence to terminate with a jump to address K in the SCM field, or to continue with the current program sequence, depending on the contents of register Xj. This decision will not be made, and the instruction will not issue from the CIW register, until the Xj register is free. The branch to address K will occur only on the conditions listed below. The current program sequence will be continued for all other cases.

Jump to K if:	(Xj) = 0000 0000 0000 0000 0000	plus zero
	(Xj) = 7777 7777 7777 7777 7777	minus zero

This instruction is intended for branching on a zero result from either a fixed point or a floating point operation.

Execution time

There are three normal cases to be considered in the execution time for this instruction. These are the following:

Branch fall through: (Minimum time two clock periods)
If the jump condition is not met this instruction is executed, and issues from the CIW register, in two clock periods. Execution will be delayed if the Xj register is not free.

Branch in stack: (Minimum time three clock periods)
If the jump condition is met and the destination address K is in the instruction stack, this instruction is executed in three clock periods. Execution will be delayed if the Xj register is not free.

Branch out of stack: (Minimum time 11 clock periods)
If the jump condition is met and the destination address K is not in the instruction stack, this instruction is executed in a minimum of 11 clock periods. Execution will be delayed if the Xj register is not free. Execution may also be delayed by bank conflicts in the SCM.

The following is the sequence of events in the execution of this instruction for the case of branch fall through.

CP00 Branch instruction in the upper parcel of the CIW register.
Instruction does not issue.
Jump condition not satisfied.
Set GJF.

CP01 Branch instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Clear GJF.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.

The following is the sequence of events in the execution of this instruction for the case of branch in stack.

CP00 Branch instruction in the upper parcel of the CIW register.
Instruction does not issue.
Jump condition satisfied.
Set GJF.

Branch instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit K to the P register.
Set JCF.
Clear GJF.

Branch instruction in the upper parcel of the CIW register.
Instruction issues.
Destination address in instruction stack.
Coincidence in the IAS.
Read a valid instruction word from IWS to the CIW register.
Clear JCF.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.

The following is the sequence of events in the execution of this instruction for the case of branch out of stack.

CP00 Branch instruction in the upper parcel of the CIW register.
Instruction does not issue.
Jump condition satisfied.
Set GJF.

CP01 Branch instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit K to the P register.
Set JCF.
Clear GJF.

CP02 Branch instruction in the upper parcel of the CIW register.
Instruction issues.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
Transmit (P) + (RAS) to the IFA register.
Set OSF.
Set JOF.
Set F1F, F2F.
Clear JCF.

CP03 No instruction in the upper parcel of the CIW register.
Transmit (P) to the NSA register.
Set M1F, M2F.
The OSF remains set.
Transmit (IFA) to the SAS. Tag for read to IWS.
Advance (IFA).
Clear JOF.
Clear F2F.

CP04 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Transmit (IFA) to the SAS. Tag for read to IWS.
Advance (IFA).
First fetch address leaves the SAS for a SCM bank.
Clear F1F.

No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
First fetch read/write cycle begins in a SCM bank.
Second fetch address leaves the SAS for a SCM bank.

No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Second fetch read/write cycle begins in a SCM bank.

No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.

No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
First fetch instruction word reads to SCM bank operand register.

CP09 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Transmit the first fetch instruction word to the IWS.
Transmit (NSA) to the IAS.
Shift the IWS and the IAS one word position.
Advance (NSA).
Second fetch instruction word reads to SCM bank operand register.
Clear M2F.

No instruction in the upper parcel of the CIW register.
Coincidence in the IAS.
Read a valid instruction word from the IWS to CIW register
Clear the OSF.
Transmit the second fetch instruction word to the IWS.
Transmit (NSA) to the IAS.
Shift the IWS and the IAS one word position.
Advance (NSA).
Clear M1F.

CP11 Next instruction in the upper parcel of the CIW register.
Instruction may issue.

Execution delays

The following delays may occur in the execution of the third sequence listed above for the case of a jump out of the instruction stack.

A delay in the execution of the instruction will occur if the F1F is set in CP02. The JOF will not set in CP02 unless the F1F is cleared, indicating all instruction fetch requests have been sent to the SAS. If the F1F is set in CP02 the rest of the sequence will be delayed, and the JOF will not set until all fetch instruction words have arrived at the IWS and the M1F has been cleared.

The transmission of (P) to the NSA register, the setting of M1F and M2F, and the clearing of JOF will not take place in CP03 if an instruction fetch is still in process and M1F is set for this earlier reference. This will not delay the execution of the remainder of this instruction. These commands will be issued and the functions performed when the previous fetch instruction word arrives at the IWS.

The transmission of (IFA) to the SAS will be delayed in CP03 or in CP04 if a backup situation exists in the SAS. The remainder of the sequence will be delayed by the amount of time required to resolve the SCM bank conflict and clear up the SAS backlog.

A delay in the arrival of the first fetch instruction word at the IWS will occur if a SCM bank conflict prevents the first fetch address from leaving the SAS in CP04. This will delay the completion of this instruction by the time required to resolve the SCM bank conflict.

Special situations

Last parcel

This instruction requires two parcels of an instruction word for normal use. If this instruction begins in the first, second, or third parcel of an instruction word the following parcel completes the instruction. If this instruction begins in the last parcel of an instruction word it will not be continued in the following word. In this case the instruction will be executed as if there were a fifth parcel in the instruction word and this parcel contained all zeros.

Previous fetch is destination address

If the branch point destination address is not in the instruction stack at the beginning of this instruction sequence, but is in process to the instruction stack as an instruction fetch address, this fetch instruction word will arrive at the IWS and enter the CIW register in less than the minimum 11 clock periods normally required for a jump out of stack. The remainder of the instruction sequence will be completed and a duplicate word will be read to the IWS as a normal initial instruction fetch. This will not cause any special problems in the IWS.

Jump out of range

If the branch point destination address is greater than the SCM field length the program range condition flag is set in the PSD register. The instruction will execute to completion, but the first instruction word for the next program sequence will not read from the IWS to the CIW register. At this point an error interrupt will occur as a result of the program range condition flag, and an exchange jump will occur to address (EEA) in the SCM. The terminating exchange package will contain the out-of-range address in the program address field.

Jump to zero

A jump to relative address zero in the SCM field is treated in the same manner as a jump out of range. The program range condition flag is set in the PSD register, and the program will be terminated with an error exit to address (EEA). The terminating exchange package will contain a zero quantity in the program address field.

Jump to breakpoint address

A jump to address (BPA) will set the breakpoint condition flag in the PSD register. The instruction will be executed to completion, and the exchange jump to address (EEA) will occur before the first instruction is executed at the branch point destination address.

I/O interrupt or error condition

If an I/O interrupt request or an error exit request exists at the time this instruction is executed, the instruction is executed to completion before the interrupt occurs.

031jx	xxxxx

Branch on X nonzero

This instruction is a two parcel instruction in which the lower order 18 bits are used as an operand K. Execution of this instruction will cause the program sequence to terminate with a jump to address K in the SCM field, or to continue with the current program sequence, depending on the contents of register Xj. This decision will not be made, and the instruction will not issue from the CIW register, until the Xj register is free. The program sequence will be continued only on the conditions listed below. The branch to address K will occur for all other cases.

Continue if: (Xj) = 0000 0000 0000 0000 0000 plus zero
(Xj) = 7777 7777 7777 7777 7777 minus zero

This instruction is intended for branching on a nonzero result from either a fixed or a floating point operation.

Execution time

There are three normal cases to be considered in the execution time for this instruction. These are the following:

Branch fall through: (Minimum time two clock periods)
Branch in stack: (Minimum time three clock periods)
Branch out of stack: (Minimum time 11 clock periods)

Details of the execution timing for this instruction are the same as those for the 030 instruction.

Execution delays

The execution delays for this instruction are the same as those listed for the 030 instruction.

Special situations

The special situations for this instruction are the same as those listed for the 030 instruction.

032jx	xxxxx

Branch on X positive

This instruction is a two parcel instruction in which the lower order 18 bits are used as an operand K. Execution of this instruction will cause the program sequence to terminate with a jump to address K in the SCM field, or to continue with the current program sequence, depending on the contents of register Xj. This decision will not be made, and the instruction will not issue from the CIW register, until the Xj register is free. The branch decision for this instruction is based on the value of the sign bit in (Xj).

Jump to K if: Bit 59 of (Xj) = 0 (positive)
Continue if: Bit 59 of (Xj) = 1 (negative)

This instruction is intended for branching on a positive result from either a fixed point or a floating point operation.

Execution time

There are three normal cases to be considered in the execution time for this instruction. These are the following:

Branch fall through: (Minimum time two clock periods)
Branch in stack: (Minimum time three clock periods)
Branch out of stack: (Minimum time 11 clock periods)

Details of the execution timing for this instruction are the same as those listed for the 030 instruction.

Execution delays

The execution delays for this instruction are the same as those listed for the 030 instruction.

Special situations

The special situations for this instruction are the same as those listed for the 030 instruction.

033jx	xxxxx

Branch on X negative

This instruction is a two parcel instruction in which the lower order 18 bits are used as an operand K. Execution of this instruction will cause the program sequence to terminate with a jump to address K in the SCM field, or to continue with the current program sequence, depending on the contents of register Xj. This decision will not be made, and the instruction will not issue from the CIW register, until the Xj register is free. The branch decision for this instruction is based on the value of the sign bit in (Xj).

Jump to K if: Bit 59 of (Xj) = 1 (negative)
Continue if: Bit 59 of (Xj) = 0 (positive)

This instruction is intended for branching on a negative result from either a fixed point or a floating point operation.

Execution time

There are three normal cases to be considered in the execution time for this instruction. These are the following:

Branch fall through: (Minimum time two clock periods)
Branch in stack: (Minimum time three clock periods)
Branch out of stack: (Minimum time 11 clock periods)

Details of the execution timing for this instruction are the same as those listed for the 030 instruction.

Execution delays

The execution delays for this instruction are the same as those listed for the 030 instruction.

Special situations

The special situations for this instruction are the same as those listed for the 030 instruction.

034jx	xxxxx

Branch on X in range

This instruction is a two parcel instruction in which the lower order 18 bits are used as an operand K. Execution of this instruction will cause the program sequence to terminate with a jump to address K in the SCM field, or to continue with the current program sequence, depending on the contents of register Xj. This decision will not be made, and the instruction will not issue from the CIW register, until the Xj register is free. The program sequence will be continued only on the conditions listed below. The branch to address K will occur for all other cases.

Continue if: (Xj) = 3777 xxxx xxxx xxxx xxxx (positive overflow)
(Xj) = 4000 xxxx xxxx xxxx xxxx (negative overflow)
(Xj) = 1777 xxxx xxxx xxxx xxxx (positive indefinite)
(Xj) = 6000 xxxx xxxx xxxx xxxx (negative indefinite)

This instruction is intended for branching on a floating point quantity within the floating point range. The value of the coefficient is ignored in making this branch test. An underflow quantity is considered in range for purposes of this branch test.

Execution time

There are three normal cases to be considered in the execution time for this instruction. These are the following:

Branch fall through: (Minimum time two clock periods)
Branch in stack: (Minimum time three clock periods)
Branch out of stack: (Minimum time 11 clock periods)

Details of the execution timing for this instruction are the same as those listed for the 030 instruction.

Execution delays

The execution delays for this instruction are the same as those listed for the 030 instruction.

Special situations

The special situations for this instruction are the same as those listed for the 030 instruction.

035jx	xxxxx

Branch on X not in range

This instruction is a two parcel instruction in which the lower order 18 bits are used as an operand K. Execution of this instruction will cause the program sequence to terminate with a jump to address K in the SCM field, or to continue with the current program sequence, depending on the contents of register Xj. This decision will not be made, and the instruction will not issue from the CIW register, until the Xj register is free. The branch to address K will occur only on the conditions listed below. The current program sequence will be continued for all other cases.

Jump to K if:
(Xj) = 3777 xxxx xxxx xxxx xxxx (positive overflow)
(Xj) = 4000 xxxx xxxx xxxx xxxx (negative overflow)
(Xj) = 1777 xxxx xxxx xxxx xxxx (positive indefinite)
(Xj) = 6000 xxxx xxxx xxxx xxxx (negative indefinite)

This instruction is intended for branching on a floating point quantity which is not in the floating point range. The value of the coefficient is ignored in making this branch test. An underflow quantity is considered in range for purposes of this branch test.

Execution time

There are three normal cases to be considered in the execution time for this instruction. These are the following:

Branch fall through: (Minimum time two clock periods)
Branch in stack: (Minimum time three clock periods)
Branch out of stack: (Minimum time 11 clock periods)

Details of the execution timing for this instruction are the same as those listed for the 030 instruction.

Execution delays

The execution delays for this instruction are the same as those listed for the 030 instruction.

Special situations

The special situations for this instruction are the same as those listed for the 030 instruction.

036jx	xxxxx

Branch on X definite

This instruction is a two parcel instruction in which the lower order 18 bits are used as an operand K. Execution of this instruction will cause the program sequence to terminate with a jump to address K in the SCM field, or to continue with the current program sequence, depending on the contents of register Xj. This decision will not be made, and the instruction will not issue from the CIW register, until the Xj register is free. The program sequence will be continued only on the conditions listed below. The branch to address K will occur for all other cases.

Continue if: (Xj) = 1777 xxxx xxxx xxxx xxxx (positive indefinite)
(Xj) = 6000 xxxx xxxx xxxx xxxx (negative indefinite)

This instruction is intended for branching on a floating point quantity which may be out of range but is still defined. The value of the coefficient is ignored in making this branch test. An overflow quantity or an underflow quantity is considered defined for purposes of this branch test.

Execution time

There are three normal cases to be considered in the execution time for this instruction. These are the following:

Branch fall through: (Minimum time two clock periods)
Branch in stack: (Minimum time three clock periods)
Branch out of stack: (Minimum time 11 clock periods)

Details of the execution timing for this instruction are the same as those listed for the 030 instruction.

Execution delays

The execution delays for this instruction are the same as those listed for the 030 instruction.

Special situations

The special situations for this instruction are the same as those listed for the 030 instruction.

037jx	xxxxx

Branch on X indefinite

This instruction is a two parcel instruction in which the lower order 18 bits are used as an operand K. Execution of this instruction will cause the program sequence to terminate with a jump to address K in the SCM field, or to continue with the current program sequence, depending on the contents of register Xj. This decision will not be made, and the instruction will not issue from the CIW register, until the Xj register is free. The branch to address K will occur only on the conditions listed below. The current program sequence will be continued for all other cases.

Jump to K if: (Xj) = 1777 xxxx xxxx xxxx xxxx (positive indefinite)
(Xj) = 6000 xxxx xxxx xxxx xxxx (negative indefinite)

This instruction is intended for branching on a floating point quantity which is not defined. The value of the coefficient is ignored in making this branch test. An overflow quantity or an underflow quantity is considered defined for purposes of this branch test.

Execution time

There are three normal cases to be considered in the execution time for this instruction. These are the following:

Branch fall through: (Minimum time two clock periods)
Branch in stack: (Minimum time three clock periods)
Branch out of stack: (Minimum time 11 clock periods)

Details of the execution timing for this instruction are the same as those listed for the 030 instruction.

Execution delays

The execution delays for this instruction are the same as those listed for the 030 instruction.

Special situations

The special situations for this instruction are the same as those listed for the 030 instruction.

04ijx	xxxxx

Branch on B .EQ. B

This instruction is a two parcel instruction in which the lower order 18 bits are used as an operand K. Execution of this instruction will cause the program sequence to terminate with a jump to address K in the SCM field, or to continue with the current program sequence, depending on a comparison of the contents of register Bi with the contents of register Bj. This decision will not be made, and the instruction will not issue from the CIW register, until the Bi register and Bj register are free. The branch to address K will occur only if the two quantities are identical on a bit by bit comparison basis. The current program sequence will be continued for all other cases.

This instruction is intended for branching on an index equality test. A quantity consisting of all zeros and a quantity consisting of all ones are not equal for this test.

Execution time

There are three normal cases to be considered in the execution time for this instruction. These are the following:

Branch fall through: (Minimum time two clock periods)
If the jump condition is not met this instruction is executed, and issues from the CIW register, in two clock periods. Execution will be delayed if the Bi register or Bj register is not free.

Branch in stack: (Minimum time three clock periods)
If the jump condition is met and the destination address K is in the instruction stack, this instruction is executed in three clock periods. Execution will be delayed if the Bi register or Bj register is not free.

Branch out of stack: (Minimum time 11 clock periods)
If the jump condition is met and the destination address K is not in the instruction stack, this instruction is executed in a minimum of 11 clock periods. Execution will be delayed if the Bi register or Bj register is not free. Execution may also be delayed by bank conflicts in the SCM.

The following is the sequence of events in the execution of this instruction for the case of branch fall through.

CP00 Branch instruction in the upper parcel of the CIW register.
Instruction does not issue.
Jump condition not satisfied.
Bi register free.
Bj register free.
Set GJF.

Branch instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit next instruction to upper parcel of CIW register.
Clear GJF.

CP02 Next instruction in the upper parcel of the CIW register.
Instruction may issue.

The following is the sequence of events in the execution of this instruction for the case of branch in stack.

CP00 Branch instruction in the upper parcel of the CIW register
Instruction does not issue.
Jump condition satisfied.
Bi register free.
Bj register free.
Set GJF.

Branch instruction in the upper parcel of the CIW register.
Instruction does not issue.
Transmit K to the P register.
Set JCF.
Clear GJF.

Branch instruction in the upper parcel of the CIW register.
Instruction issues.
Destination address in instruction stack.
Coincidence in the IAS.
Read a valid instruction word from IWS to the CIW register.
Clear JCF.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.

The following is the sequence of events in the execution of this instruction for the case of branch out of stack.

```
      Branch instruction in the upper parcel of the CIW register.
      Instruction does not issue.
      Jump condition satisfied.
      Bi register free.
      Bj register free.
      Set GJF.

      Branch instruction in the upper parcel of the CIW register.
      Instruction does not issue.
      Transmit K to the P register.
      Set JCF.
      Clear GJF.

CP02  Branch instruction in the upper parcel of the CIW register.
      Instruction issues.
      No coincidence in the IAS.
      Read a blank word from the IWS to the CIW register.
      Transmit (P) + (RAS) to the IFA register.
      Set OSF.
      Set JOF.
      Set F1F, F2F.
      Clear JCF.

CP03  No instruction in the upper parcel of the CIW register.
      Transmit (P) to the NSA register.
      Set M1F, M2F.
      The OSF remains set.
      Transmit (IFA) to the SAS.  Tag for read to IWS.
      Advance (IFA).
      Clear JOF.
      Clear F2F.

      No instruction in the upper parcel of the CIW register.
      No coincidence in the IAS.
      Read a blank word from the IWS to the CIW register.
      The OSF remains set.
      Transmit (IFA) to the SAS.  Tag for read to IWS.
      Advance (IFA).
      First fetch address leaves the SAS for a SCM bank.
      Clear F1F.
```

CP05 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
First fetch read/write cycle begins in a SCM bank.
Second fetch address leaves the SAS for a SCM bank.

No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Second fetch read/write cycle begins in a SCM bank.

CP07 No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.

No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
First fetch instruction word reads to SCM bank operand register.

No instruction in the upper parcel of the CIW register.
No coincidence in the IAS.
Read a blank word from the IWS to the CIW register.
The OSF remains set.
Transmit the first fetch instruction word to the IWS.
Transmit (NSA) to the IAS.
Shift the IWS and the IAS one word position.
Advance (NSA).
Second fetch instruction word reads to SCM bank operand register.
Clear M2F.

CP10 No instruction in the upper parcel of the CIW register.
Coincidence in the IAS.
Read a valid instruction word from the IWS to CIW register.
Clear the OSF.
Transmit the second fetch instruction word to the IWS.
Transmit (NSA) to the IAS.
Shift the IWS and the IAS one word position.
Advance (NSA).
Clear M1F.

CP11 Next instruction in the upper parcel of the CIW register. Instruction may issue.

Execution delays

The following delays may occur in the execution of the third sequence listed above for the case of a jump out of the instruction stack.

A delay in the execution of the instruction will occur if the F1F is set in CP02. The J0F will not set in CP02 unless the F1F is cleared, indicating all instruction fetch requests have been sent to the SAS. If the F1F is set in CP02 the rest of the sequence will be delayed, and the J0F will not set until all fetch instruction words have arrived at the IWS and the M1F has been cleared.

The transmission of (P) to the NSA register, the setting of M1F and M2F, and the clearing of J0F will not take place in CP03 if an instruction fetch is still in process and M1F is set for this earlier reference. This will not delay the execution of the remainder of this instruction. These commands will be issued and the functions performed when the previous fetch instruction word arrives at the IWS.

The transmission of (IFA) to the SAS will be delayed in CP03 or in CP04 if a backup situation exists in the SAS. The remainder of the sequence will be delayed by the amount of time required to resolve the SCM bank conflict and clear up the SAS backlog.

A delay in the arrival of the first fetch instruction word at the IWS will occur if a SCM bank conflict prevents the first fetch address from leaving the SAS in CP04. This will delay the completion of this instruction by the time required to resolve the SCM bank conflict.

Special situations

Designators i and j have same value

The i and j designators may have the same value in this instruction. In this case the designated B register is compared against itself. The branch condition test is made in the same manner for this case as would be made if two different B registers were designated and the contents of the two B registers were identical.

Last parcel

This instruction requires two parcels of an instruction word for normal use. If this instruction begins in the first, second, or third parcel of an instruction word the following parcel completes the instruction. If this instruction begins in the last parcel of an instruction word it will not be continued in the following word In this case the instruction will be executed as if there were a fifth parcel in the instruction word and this parcel contained all zeros.

Previous fetch is destination address

If the branch point destination address is not in the instruction stack at the beginning of this instruction sequence, but is in process to the instruction stack as an instruction fetch address, this fetch instruction word will arrive at the IWS and enter the CIW register in less than the minimum 11 clock periods normally required for a jump out of stack. The remainder of the instruction sequence will be completed and a duplicate word will be read to the IWS as a normal initial instruction fetch. This will not cause any special problems in the IWS.

Jump out of range

If the branch point destination address is greater than the SCM field length the program range condition flag is set in the PSD register. The instruction will execute to completion, but the first instruction word for the next program sequence will not read from the IWS to the CIW register. At this point an error interrupt will occur as a result of the program range condition flag, and an exchange jump will occur to address (EEA) in the SCM. The terminating exchange package will contain the out-of-range address in the program address field.

Jump to zero

A jump to relative address zero in the SCM field is treated in the same manner as a jump out of range. The program range condition flag is set in the PSD register, and the program will be terminated with an error exit to address (EEA). The terminating exchange package will contain a zero quantity in the program address field.

Jump to breakpoint address

A jump to address (BPA) will set the breakpoint condition flag in the PSD register. The instruction will be executed to completion, and the exchange jump to address (EEA) will occur before the first instruction is executed at the branch point destination address.

I/O interrupt or error condition

If an I/O interrupt request or an error exit request exists at the time this instruction is executed, the instruction is executed to completion before the interrupt occurs.

05ijx	xxxxx

Branch on B .NE. B

This instruction is a two parcel instruction in which the lower order 18 bits are used as an operand K. Execution of this instruction will cause the program sequence to terminate with a jump to address K in the SCM field, or to continue with the current program sequence, depending on a comparison of the contents of register Bi with the contents of register Bj. This decision will not be made, and the instruction will not issue from the CIW register, until the Bi register and Bj register are free. The program sequence will be continued only if the two quantities are identical on a bit by bit comparison basis. The branch to address K will occur for all other cases.

This instruction is intended for branching on an index inequality test. A quantity consisting of all zeros and a quantity consisting of all ones are not equal for this test.

Execution time

There are three normal cases to be considered in the execution time for this instruction. These are the following:

Branch fall through: (Minimum time two clock periods)
Branch in stack: (Minimum time three clock periods)
Branch out of stack: (Minimum time 11 clock periods)

Details of the execution timing for this instruction are the same as those for the 04 instruction.

Execution delays

The execution delays for this instruction are the same as those listed for the 04 instruction.

Special situations

The special situations for this instruction are the same as those listed for the 04 instruction.

06ijx	xxxxx

Branch on B .GE. B

This instruction is a two parcel instruction in which the lower order 18 bits are used as an operand K. Execution of this instruction will cause the program sequence to terminate with a jump to address K in the SCM field, or to continue with the current program sequence, depending on a comparison of the contents of register Bi with the contents of register Bj. Both quantities are treated as signed integers. This decision will not be made, and the instruction will not issue from the CIW register, until the Bi register and Bj register are free. The branch to address K will occur if the content of register Bi is greater than, or equal to, the content of register Bj. The current program sequence will be continued if the content of register Bi is less than the content of register Bj.

This instruction is intended for branching on an index threshold test. The test is made in a 19 bit ones complement mode. The quantity (Bi) and the quantity (Bj) are sign extended one bit to prevent an erroneous result caused by exceeding the modulus of the comparison device. The quantity (Bj) is then subtracted from the quantity (Bi). The branch decision is based on the sign bit in the 19 bit result. A branch to address K occurs if the sign of the result is positive. The current sequence is continued if the sign of the result is negative. A positive zero quantity and a negative zero quantity are not treated as equal in this test. The four possible combinations of positive and negative zero values are summarized below.

Jump to K if:	(Bi) = 000000	and	(Bj) = 000000
	(Bi) = 777777	and	(Bj) = 777777
	(Bi) = 000000	and	(Bj) = 777777
Continue if:	(Bi) = 777777	and	(Bj) = 000000

Execution time

Execution delays

Special situations

Details of the execution timing, execution delays, and special situations for this instruction are the same as those listed for the 04 instruction.

07ijx	xxxxx

Branch on B .LT. B

This instruction is a two parcel instruction in which the lower order 18 bits are used as an operand K. Execution of this instruction will cause the program sequence to terminate with a jump to address K in the SCM field, or to continue with the current program sequence, depending on a comparison of the contents of register Bi with the contents of register Bj. Both quantities are treated as signed integers. This decision will not be made, and the instruction will not issue from the CIW register, until the Bi register and Bj register are free. The branch to address K will occur if the content of register Bi is less than the content of register Bj. The current program sequence will be continued if the content of register Bi is greater than, or equal to, the content of register Bj.

This instruction is intended for branching on an index threshold test. The test is made in a 19 bit ones complement mode. The quantity (Bi) and the quantity (Bj) are sign extended one bit to prevent an erroneous result caused by exceeding the modulus of the comparison device. The quantity (Bj) is then subtracted from the quantity (Bi). The branch decision is based on the sign bit in the 19 bit result. A branch to address K occurs if the sign of the result is negative. The current sequence is continued if the sign of the result is positive. A positive zero quantity and a negative zero quantity are not treated as equal in this test. The four possible combinations of positive and negative zero values are summarized below.

Jump to K if:	(Bi) = 777777	and	(Bj) = 000000
Continue if:	(Bi) = 000000	and	(Bj) = 000000
	(Bi) = 777777	and	(Bj) = 777777
	(Bi) = 000000	and	(Bj) = 777777

Execution time

Execution delays

Special situations

Details of the execution timing, execution delays, and special situations for this instruction are the same as those listed for the 04 instruction.

10ij0 Copy

This instruction causes the boolean unit to read a 60 bit word from register Xj and copy this word into register Xi.

This instruction is intended for moving data from one X register to another X register as rapidly as possible. No logical function is performed on the data.

Issue conditions

Xi register is free.
Xj register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 10 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the boolean unit.
Set Xi reservation flag.
Set go boolean flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from boolean unit to Xd register.
Clear Xd reservation flag.
Clear go boolean flag.

Special situations

Designator k not zero

The k designator in this instruction is normally zero. This designator is ignored, however, in the execution of the instruction, and a nonzero value will have no effect on the results.

Designators i and j have the same value

If the i and j designators have the same value this instruction will read a 60 bit word from the designated X register and then write the same information back into that X register. The timing for this case will be the same as the timing for the general case, and no special conflicts will occur.

11ijk Logical product

This instruction causes the boolean unit to read operands from two X registers, operate upon them to form a single word result, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). The resultant word delivered to the Xi register is the bit by bit logical product of the two operands. Each of the 60 bits in (Xj) is acted upon by the corresponding bit of (Xk) to form a single bit in (Xi). A sample computation is listed below in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

```
Sample operands:  (Xj) = 7777 7000 0123 4567 1010
                  (Xk) = 0123 4567 0077 7700 1100
                  (Xi) = 0123 4000 0023 4500 1000
```

This instruction is intended for extracting portions of a 60 bit word during data processing as distinguished from numerical computation. This instruction together with the other boolean and shift instructions may be used to manipulate alphanumeric or other coded data not related to the 60 bit machine word length.

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 11 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the boolean unit.
Transmit (Xk) to the boolean unit.
Set Xi reservation flag.
Set go boolean flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from boolean unit to Xd register.
Clear Xd reservation flag.
Clear go boolean flag.

Special situations

Designators j and k have the same value

If the j and k designators have the same value in this instruction the designated X register content is operated upon by a copy of this same quantity. The instruction in this case degenerates into a copy instruction. The timing for this case will be the same as the timing for the general case, and no special conflicts will occur.

Designators i and j have the same value

If the i and j designators have the same value in this instruction the quantity (Xj) is replaced by the resultant quantity (Xi) at the end of the operation. No special conflicts will occur as a result of this combination.

Designators i and k have the same value

If the i and k designators have the same value in this instruction the quantity (Xk) is replaced by the resultant quantity (Xi) at the end of the operation. No special conflicts will occur as a result of this combination.

12ijk Logical sum

This instruction causes the boolean unit to read operands from two X registers, operate upon them to form a single word result, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). The resultant word delivered to the Xi register is the bit by bit logical sum of the two operands. Each of the 60 bits in (Xj) is acted upon by the corresponding bit of (Xk) to form a single bit in (Xi). A sample computation is listed below in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

Sample operands:
(Xj) = 0000 7777 0123 4567 1010
(Xk) = 0123 4567 7777 0000 1100
(Xi) = 0123 7777 7777 4567 1110

This instruction is intended for merging portions of a 60 bit word into a composite word during data processing as distinguished from numerical computation. This instruction together with the other boolean and shift instructions may be used to manipulate alphanumeric or other coded data not related to the 60 bit machine word length.

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 12 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the boolean unit.
Transmit (Xk) to the boolean unit.
Set Xi reservation flag.
Set go boolean flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from boolean unit to Xd register.
Clear Xd reservation flag.
Clear go boolean flag.

Special situations

Designators j and k have the same value

If the j and k designators have the same value in this instruction the designated X register content is merged with another copy of the same quantity. The instruction in this case degenerates into a copy instruction. The timing for this case will be the same as the timing for the general case, and no special conflicts will occur.

Designators i and j have the same value

If the i and j designators have the same value in this instruction the quantity (Xj) is replaced by the resultant quantity (Xi) at the end of the operation. No special conflicts will occur as a result of this combination.

Designators i and k have the same value

If the i and k designators have the same value in this instruction the quantity (Xk) is replaced by the resultant quantity (Xi) at the end of the operation. No special conflicts will occur as a result of this combination.

13ijk Logical difference

This instruction causes the boolean unit to read operands from two X registers, operate upon them to form a single word result, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). The resultant word delivered to the Xi register is the bit by bit logical difference of the two operands. Each of the 60 bits in (Xj) is acted upon by the corresponding bit of (Xk) to form a single bit in (Xi). A sample computation is listed below in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

```
Sample operands:  (Xj) = 0123 7777 0123 4567 1010
                  (Xk) = 0123 4567 7777 3210 1100
                  (Xi) = 0000 3210 7654 7777 0110
```

This instruction is intended for comparing bit patterns or for complementing bit patterns during data processing as distinguished from numerical computation. This instruction together with the other boolean and shift instructions may be used to manipulate alphanumeric or other coded data not related to the 60 bit machine word length.

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 13 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the boolean unit.
Transmit (Xk) to the boolean unit.
Set Xi reservation flag.
Set go boolean flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from boolean unit to Xd register.
Clear Xd reservation flag.
Clear go boolean flag.

Special situations

Designators j and k have the same value

If the j and k designators have the same value in this instruction a logical difference is formed between two identical quantities. The result will be a word of all zeros written into register Xi. The timing for this case is the same as the timing for the general case.

Designators i and j have the same value

If the i and j designators have the same value in this instruction the quantity (Xj) is replaced by the resultant quantity (Xi) at the end of the operation. No special conflicts will occur as a result of this combination.

Designators i and k have the same value

If the i and k designators have the same value in this instruction the quantity (Xk) is replaced by the resultant quantity (Xi) at the end of the operation. No special conflicts will occur as a result of this combination.

14i0k Copy complement

This instruction causes the boolean unit to read a 60 bit word from register Xk, complement the word, and write the result into register Xi.

This instruction is intended for changing the sign of a fixed point or floating point quantity as quickly as possible. This instruction is also useful in data processing for inverting an entire 60 bit field. The result is generally, but not necessarily, returned to the same X register.

Issue conditions

Xi register is free.
Xk register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

14 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xk) to the boolean unit.
Set Xi reservation flag.
Set go boolean flag.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from boolean unit to Xd register.
Clear Xd reservation flag.
Clear go boolean flag.

Special situations

Designator j not zero

The j designator in this instruction is normally zero. This designator is ignored, however, in the execution of the instruction, and a nonzero value will have no effect on the results.

Designators i and k have the same value

The i and k designators will frequently have the same value in this instruction. In this case the quantity read from the designated X register is complemented and returned to the same X register. The timing for this case is the same as the timing for the general case.

15ijk Logical product with complement

This instruction causes the boolean unit to read operands from two X registers, operate upon them to form a single word result, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). The resultant word delivered to the Xi register is the bit by bit logical product of (Xj) with the complement of (Xk). Each of the 60 bits in (Xj) is acted upon by the corresponding bit of (Xk) to form a single bit in (Xi). A sample computation is listed below in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

Sample operands:
(Xj) = 7777 7000 0123 4567 1010
(Xk) = 0123 4567 0007 7700 1100
(Xi) = 7654 3000 0120 0067 0010

This instruction is intended for extracting portions of a 60 bit word during data processing as distinguished from numerical computation. This instruction together with the other boolean and shift instructions may be used to manipulate alphanumeric or other coded data not related to the 60 bit machine word length.

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 15 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the boolean unit.
Transmit (Xk) to the boolean unit.
Set Xi reservation flag.
Set go boolean flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from boolean unit to Xd register.
Clear Xd reservation flag.
Clear go boolean flag.

Special situations

Designators j and k have the same value

If the j and k designators have the same value in this instruction a logical product is formed between two complementary quantities. The result will be a word of all zeros written into register Xi. The timing for this case is the same as the timing for the general case.

Designators i and j have the same value

If the i and j designators have the same value in this instruction the quantity (Xj) is replaced by the resultant quantity (Xi) at the end of the operation. No special conflicts will occur as a result of this combination.

Designators i and k have the same value

If the i and k designators have the same value in this instruction the quantity (Xk) is replaced by the resultant quantity (Xi) at the end of the operation. No special conflicts will occur as a result of this combination.

16ijk Logical sum with complement

This instruction causes the boolean unit to read operands from two X registers, operate upon them to form a single word result, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). The resultant word delivered to the Xi register is the bit by bit logical sum of (Xj) with the complement of (Xk). Each of the 60 bits in (Xj) is acted upon by the corresponding bit of (Xk) to form a single bit in (Xi). A sample computation is listed below in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

Sample operands:
(Xj) = 0000 7777 0123 4567 1010
(Xk) = 0123 4567 7777 0000 1100
(Xi) = 7654 7777 0123 7777 1011

This instruction is intended for merging portions of a 60 bit word into a composite word during data processing as distinguished from numerical computation. This instruction together with the other boolean and shift instructions may be used to manipulate alphanumeric or other coded data not related to the 60 bit machine word length.

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free in next clock period
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 16 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the boolean unit.
Transmit (Xk) to the boolean unit.
Set Xi reservation flag.
Set go boolean flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from boolean unit to Xd register.
Clear Xd reservation flag.
Clear go boolean flag.

Special situations

Designators j and k have the same value

If the j and k designators have the same value in this instruction a logical sum is formed from two complementary quantities. The result will be a word of all ones written into register Xi. The timing for this case is the same as the timing for the general case.

Designators i and j have the same value

If the i and j designators have the same value in this instruction the quantity (Xj) is replaced by the resultant quantity (Xi) at the end of the operation. No special conflicts will occur as a result of this combination.

Designators i and k have the same value

If the i and k designators have the same value in this instruction the quantity (Xk) is replaced by the resultant quantity (Xi) at the end of the operation. No special conflicts will occur as a result of this combination.

17ijk Logical difference with complement

This instruction causes the boolean unit to read operands from two X registers, operate upon them to form a single word result, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). The resultant word delivered to the Xi register is the bit by bit logical difference of (Xj) with the complement of (Xk). Each of the 60 bits in (Xj) is acted upon by the corresponding bit of (Xk) to form a single bit in (Xi). A sample computation is listed below in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

```
Sample operands:  (Xj) = 0123 7777 0123 4567 1010
                  (Xk) = 0123 4567 7777 3210 1100
                  (Xi) = 7777 4567 0123 0000 1001
```

This instruction is intended for comparing bit patterns or for complementing bit patterns during data processing as distinguished from numerical computation. This instruction together with the other boolean and shift instructions may be used to manipulate alphanumeric or other coded data not related to the 60 bit machine word length.

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 17 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the boolean unit.
Transmit (Xk) to the boolean unit.
Set Xi reservation flag.
Set go boolean flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from boolean unit to Xd register.
Clear Xd reservation flag.
Clear go boolean flag.

Special situations

Designators j and k have the same value

If the j and k designators have the same value in this instruction a logical difference is formed between two complementary quantities. The result will be a word of all ones written into register Xi. The timing for this case is the same as the timing for the general case.

Designators i and j have the same value

If the i and j designators have the same value in this instruction the quantity (Xj) is replaced by the resultant quantity (Xi) at the end of the operation. No special conflicts will occur as a result of this combination.

Designators i and k have the same value

If the i and k designators have the same value in this instruction the quantity (Xk) is replaced by the resultant quantity (Xi) at the end of the operation. No special conflicts will occur as a result of this combination.

20ijk Left shift X by jk

This instruction causes the shift unit to read one operand from the Xi register, shift the 60 bit word left circularly by jk bit positions, and then write the resulting 60 bit word back into the same Xi register. The designators j and k are treated as a single six bit positive integer operand in this instruction.

A left circular shift implies that the bit pattern in the 60 bit word is displaced toward the higher order bit positions. The bits which are shifted off the upper end of the 60 bit word are inserted in the lowest order bit positions in the same sequence. The resulting 60 bit word has the same quantity of bits with a value of one, and the same quantity with a value of zero, as in the original operand.

A sample computation is listed below in octal notation to illustrate the operation performed. The j designator has a value of 1 and the k designator a value of 2 in this example. These octal quantities are treated as a shift count of 12 octal, or 10 decimal.

Sample operands: Initial (Xi) = 2323 6600 0000 0000 0111
jk = 12
Terminal (Xi) = 7540 0000 0000 0022 2464

This instruction is intended for use in data processing as distinguished from numerical computation. This instruction, together with the companion instruction 21, may be used whenever a data word is to be shifted by a predetermined amount. If the amount of shift is derived in the execution of the program, instruction 22 or 23 should be used.

Issue conditions

Xi register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 20 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xi) to the shift unit.
Set Xi reservation flag.
Set go shift flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from shift unit to Xd register.
Clear Xd reservation flag.
Clear go shift flag.

Special situations

Shift count is zero

If the j and k designators are zero this instruction reads the operand from register Xi and returns the result unaltered to the same register. The timing for this case is the same as the timing for the general case.

Shift count is greater than 60

If the shift count is greater than the 60 bit register length the shift is performed modulo 60. For example, if the shift count is 63 (decimal) the result is a three bit position shift.

Operand all ones or all zeros

An all ones or all zeros word is treated in the same manner as any other bit pattern. The timing for this case is the same as the timing for the general case.

21ijk	Right shift X by jk

This instruction causes the shift unit to read one operand from the Xi register, shift the 60 bit word right with sign extension by jk bit positions, and then write the resulting 60 bit word back into the same Xi register. The designators j and k are treated as a single six bit positive integer operand in this instruction.

A right shift with sign extension implies that the bit pattern in the 60 bit word is displaced toward the lower order bit positions. The bits which are shifted off the lower end of the word are discarded. The highest order bit positions are filled with copies of the original sign bit.

Two sample computations are listed below in octal notation to illustrate the operation performed. The first example contains a positive operand and the second example a negative operand.

```
Sample operands:  Initial (Xi)  = 2004 7655 0002 3400 0004
                  jk = 30 (octal)
                  Terminal (Xi) = 0000 0000 2004 7655 0002

                  Initial (Xi)  = 6000 4420 2222 0000 5643
                  jk = 10 (octal)
                  Terminal (Xi) = 7774 0011 0404 4440 0013
```

This instruction may be used whenever a data word is to be shifted right with sign extension by a predetermined amount. If the amount of shift is derived in the execution of the program, instruction 22 or 23 should be used.

Issue conditions

Xi register is free.
X register input path will be free in next clock period
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

21 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xi) to the shift unit.
Set Xi reservation flag.
Set go shift flag.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from shift unit to Xd register.
Clear Xd reservation flag.
Clear go shift flag.

Special situations

Shift count is zero

If the j and k designators are zero this instruction reads the operand from register Xi and returns the result unaltered to the same register. The timing for this case is the same as the timing for the general case.

Shift count is greater than 60

If the shift count is greater than the 60 bit register length the resulting word will contain 60 copies of the sign bit. If the operand was positive, a positive zero word will result. If the operand was negative, a negative zero word will result.

Operand all ones or all zeros

An all ones or all zeros word is treated in the same manner as any other bit pattern. The timing for this case is the same as the timing for the general case.

22ijk Left shift X by B

This instruction causes the shift unit to read a 60 bit operand from the Xk register, shift the data either left or right as specified by (Bj), and then write the resulting 60 bit word into the Xi register. If (Bj) is positive the data is shifted to the left in a circular mode the number of bit positions designated by (Bj). If (Bj) is negative the data is shifted to the right with sign extension the number of bit positions designated by the magnitude of (Bj).

A left circular shift implies that the bit pattern in the 60 bit word is displaced toward the higher order bit positions. The bits which are shifted off the upper end of the 60 bit word are inserted in the lowest order bit positions in the same sequence. The resulting 60 bit word has the same quantity of bits with a value of one, and the same quantity with a value of zero, as in the original operand.

A right shift with sign extension implies that the bit pattern in the 60 bit word is displaced toward the lower order bit positions. The bits which are shifted off the lower end of the word are discarded. The highest order bit positions are filled with copies of the original sign bit.

Three sample computations are listed below in octal notation to illustrate the operation performed. The first example contains a positive shift count resulting in a left circular shift. The last two examples illustrate the right shift with sign extension.

Sample operands:
(Xk) = 2323 6600 0000 0000 0111
(Bj) = 00 0012
(Xi) = 7540 0000 0000 0022 2464

(Xk) = 1327 6000 0000 3333 2422
(Bj) = 77 7771
(Xi) = 0013 2760 0000 0033 3324

(Xk) = 5327 6000 0000 3333 2422
(Bj) = 77 7771
(Xi) = 7753 2760 0000 0033 3324

This instruction is intended for use in data processing where the amount of shift is derived in the computation. This instruction is also useful for correcting the coefficient of a floating point number when the exponent has been unpacked into a B register.

Issue conditions

Xi register is free.
Bj register is free.
Xk register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 22 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xk) to the shift unit.
Transmit (Bj) to the shift unit.
Set Xi reservation flag.
Set go shift flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from shift unit to Xd register.
Clear Xd reservation flag.
Clear go shift flag.

Special situations

(Bj) is zero

If (Bj) is zero, either 000000 or 777777, this instruction reads the operand from the Xk register and copies it unaltered into the Xi register. The timing for this case is the same as the timing for the general case.

(Bj) positive with magnitude greater than 60 decimal

If (Bj) is positive only the lowest order six bits are used in determining the shift count. The higher order bits are ignored. The resulting six bit shift count is treated modulo 60 decimal. For example, a shift count of 63 decimal results in a left circular shift of three bit positions.

(Bj) negative with magnitude greater than 60 decimal

If (Bj) is negative only the lowest order 12 bits are used in determining the shift count. The higher order bits are ignored. The lowest order 12 bits of (Bj) are complemented, and the resulting positive integer determines the shift count. If this shift count is greater than 60 decimal the resulting word stored in the Xi register will consist of 60 copies of the original operand sign bit.

Operand all ones or all zeros

An all ones or all zeros word is treated in the same manner as any other bit pattern. The timing for this case is the same as the timing for the general case.

23ijk Right shift X by B

This instruction causes the shift unit to read a 60 bit operand from the Xk register, shift the data either left or right as specified by (Bj), and then write the resulting 60 bit word into the Xi register. If (Bj) is positive the data is shifted to the right with sign extension the number of bit positions designated by (Bj). If (Bj) is negative the data is shifted to the left in a circular mode the number of bit positions designated by the magnitude of (Bj).

A left circular shift implies that the bit pattern in the 60 bit word is displaced toward the higher order bit positions. The bits which are shifted off the upper end of the 60 bit word are inserted in the lowest order bit positions in the same sequence. The resulting 60 bit word has the same quantity of bits with a value of one, and the same quantity with a value of zero, as in the original operand.

A right shift with sign extension implies that the bit pattern in the 60 bit word is displaced toward the lower order bit positions. The bits which are shifted off the lower end of the word are discarded. The highest order bit positions are filled with copies of the original sign bit.

Three sample computations are listed below in octal notation to illustrate the operation performed. The first two examples contain a positive shift count and result in a right shift with sign extension. The last example contains a negative shift count and results in a left circular shift.

```
Sample operands:  (Xk) = 1327 6000 0000 3333 2422
                  (Bj) = 00 0006
                  (Xi) = 0013 2760 0000 0033 3324

                  (Xk) = 5327 6000 0000 3333 2422
                  (Bj) = 00 0006
                  (Xi) = 7753 2760 0000 0033 3324

                  (Xk) = 2323 6600 0000 0000 0111
                  (Bj) = 77 7765
                  (Xi) = 7540 0000 0000 0022 2464
```

This instruction is intended for use in data processing where the amount of shift is derived in the computation. This instruction is also useful for correcting the coefficient of a floating point number when the exponent has been unpacked into a B register.

Issue conditions

Xi register is free.
Bj register is free.
Xk register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 23 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xk) to the shift unit.
Transmit (Bj) to the shift unit.
Set Xi reservation flag.
Set go shift flag.

CP01 Next instruction in the upper parcel of the CIW register
Instruction may issue.
Transmit data from shift unit to Xd register.
Clear Xd reservation flag.
Clear go shift flag.

Special situations

(Bj) is zero

If (Bj) is zero, either 000000 or 777777, this instruction reads the operand from the Xk register and copies it unaltered into the Xi register. The timing for this case is the same as the timing for the general case.

(Bj) positive with magnitude greater than 60 decimal

If (Bj) is positive only the lowest order 12 bits are used in determining the shift count. The higher order bits are ignored. If this resulting 12 bit shift count is greater than 60 decimal the resulting word stored in the Xi register will consist of 60 copies of the original operand sign bit.

(Bj) negative with magnitude greater than 60 decimal

If (Bj) is negative only the lowest order six bits are used in determining the shift count. The higher order bits are ignored. The lowest order six bits of (Bj) are complemented, and the resulting positive integer shift count is treated modulo 60 decimal. For example, a shift count of 63 decimal results in a left circular shift of three bit positions.

Operand all ones or all zeros

An all ones or all zeros word is treated in the same manner as any other bit pattern. The timing for this case is the same as the timing for the general case.

24ijk Normalize X to X, B

This instruction causes the normalize unit to read one operand from the Xk register, perform a normalizing operation on this word in a floating point format, and then deliver the normalized result to the Xi register. In addition the normalize unit will deliver a positive integer shift count to the Bj register. This shift count will be the number of bit positions of shift required to normalize the original operand coefficient.

The normalizing operation performed by the normalize unit in executing this instruction consists of repositioning the coefficient portion of the operand and then adjusting the exponent portion of the operand to leave the value of the resulting word unaltered. The coefficient portion of the operand is displaced toward the higher order bit positions of the word. The coefficient is shifted the minimum number of bit positions required to make bit 47 different from the sign bit 59. This places the most significant bit of the coefficient in the highest order bit position of the coefficient portion of the word. The exponent portion of the word is then decreased by the number of bit positions shifted.

Two sample computations are listed below in octal notation to illustrate the operation performed. The first example involves a positive floating point number and the second example a negative number.

Sample operands:
(Xk) = 2034 0047 6500 0000 2262
(Xi) = 2026 4765 0000 0022 6200
(Bj) = 00 0006

(Xk) = 5743 7730 1277 7777 5515
(Xi) = 5751 3012 7777 7755 1577
(Bj) = 00 0006

This instruction is intended for use in normalized floating point computation in which rounding is not desired. If rounding is desired the 25 instruction should be used.

Issue conditions

Xi register is free.
Xk register is free.
Bj register is free.
X register input path will be free two clock periods hence.
B register input path will be free two clock periods hence.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The results will be delivered to the Xi register and the Bj register two clock periods after the instruction issues. The Xi register and the Bj register will be reserved for the two clock periods from issue to delivery of data. The command timing for this instruction is listed below.

CP00 24 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xk) to the normalize unit.
Set Xi reservation flag.
Set Bj reservation flag.
Set go normalize flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Calculate the normalize shift count.
Move operand from input register to internal register.
Clear go normalize flag.

CP02 Transmit data from normalize unit to Xd register.
Transmit data from normalize unit to Be register.
Clear Xd reservation flag.
Clear Be reservation flag.

Special situations

Special case operand

The normalize unit makes a special case test on (Xk) at the beginning of execution for this instruction. If one of these special cases is present the operand is copied unaltered to the Xi register. The shift count entered in the Bj register is zero for these cases. There are no condition flags set in the PSD register by the normalize unit. The special case formats are listed below and consist of partial overflow, complete overflow, and indefinite forms.

Special case formats:
(Xk) = 3777 xxxx xxxx xxxx xxxx
(Xk) = 4000 xxxx xxxx xxxx xxxx
(Xk) = 1777 xxxx xxxx xxxx xxxx
(Xk) = 6000 xxxx xxxx xxxx xxxx

Complete underflow

A complete underflow occurs for this instruction whenever (Xk) is not a special case and the normalizing process results in an unpacked exponent more negative than -1777 octal. In this situation a zero word is delivered to the Xi register. The sign of the operand is preserved in this process and (Xi) is either all zero bits, or all one bits, depending on the sign of the original operand. The shift count delivered to the Bj register is a result of considering the coefficient field of (Xk) without regard to the exponent. This quantity is therefore the value which would be appropriate for normalizing the operand if the exponent were in range. There are no condition flags set in the PSD register by the normalize unit.

Partial underflow

A partial underflow occurs for this instruction whenever (Xk) is not a special case and the normalizing process results in an unpacked exponent exactly equal to -1777 octal. In this situation the result is delivered to the Xi register and the Bj register as for a normal case even though subsequent computation may detect this operand as an underflow case.

Coefficient is zero

The normalize unit in execution of this instruction treats an operand with a zero coefficient as a special underflow situation. This special situation exists for either positive or negative numbers whenever the sign bit is the same as each bit in the coefficient field of the operand and the exponent field does not qualify the operand as a special case format. There is no possibility of creating a normalized coefficient for this case. In this situation a zero word is delivered to the Xi register. The sign of the operand is preserved in this process and (Xi) is either all zero bits, or all one bits, depending on the sign of the original operand. The shift count delivered to the Bj register is 48 decimal for this case. There are no condition flags set in the PSD register by the normalize unit.

Underflow operand

A special situation exists for executing this instruction if (Xk) is in one of the two formats listed below.

Underflow formats: (Xk) = 0000 xxxx xxxx xxxx xxxx
(Xk) = 7777 xxxx xxxx xxxx xxxx

These formats include positive and negative numbers in either a partial underflow or a complete underflow form. These cases are generally covered by one of the other special situations listed above. If (Xk) is a partial underflow quantity and the coefficient is normalized, the execution of this instruction will proceed as for a normal operand and the result will be an unaltered copy of the original operand. The shift count delivered to the Bj register will be zero. If (Xk) is a partial underflow quantity and the coefficient is not normalized, a complete underflow will occur as described under that heading.

If (Xk) is a complete underflow quantity the execution of this instruction will be dominated by the fact that the coefficient is zero. This special situation is described under that heading above. The net result will be an unaltered complete underflow quantity delivered to the Xi register and a shift count of 48 decimal delivered to the Bj register.

25ijk Round normalize X to X, B

This instruction causes the normalize unit to read one operand from the Xk register, perform a rounding operation and then a normalizing operation on this word in floating point format, and finally deliver the round normalized result to the Xi register. In addition the normalize unit will deliver a positive integer shift count to the Bj register. This shift count will be the number of bit positions of shift required to normalize the original operand coefficient.

The rounding operation performed in the execution of this instruction consists of adding a bit to the coefficient portion of the operand in a bit position immediately below the least significant bit position of the original operand coefficient. This round bit has a value equal to the complement of the operand sign bit. The net result of this rounding operation is to increase the magnitude of the operand coefficient by one half.

The normalizing operation performed in the execution of this instruction consists of repositioning the coefficient and adjusting the exponent to leave the value of the resulting floating point quantity unaltered. The coefficient portion of the operand is displaced toward the higher order bit positions in the word. The round bit is shifted along with the coefficient. The displacement is the minimum number of bit positions required to make bit 47 different from the sign bit 59. This places the most significant bit of the coefficient in the highest order bit position of the coefficient portion of the word. The exponent portion of the word is decreased by the number of bit positions shifted.

Two sample computations are listed below in octal notation to illustrate the operation performed. The first example involves a positive floating point number and the second example a negative number.

Sample operands:
(Xk) = 2034 0047 6500 0000 2262
(Xi) = 2026 4765 0000 0022 6240
(Bj) = 00 0006

(Xk) = 5743 7730 1277 7777 5515
(Xi) = 5751 3012 7777 7755 1537
(Bj) = 00 0006

Issue conditions

Xi register is free.
Xk register is free.
Bj register is free.
X register input path will be free two clock periods hence.
B register input path will be free two clock periods hence.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The results will be delivered to the Xi register and the Bj register two clock periods after the instruction issues. The Xi register and the Bj register will be reserved for the two clock periods from issue to delivery of data. The command timing for this instruction is listed below.

CP00 25 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xk) to the normalize unit.
Set Xi reservation flag.
Set Bj reservation flag.
Set go normalize flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Calculate the normalize shift count.
Move operand from input register to internal register.
Clear go normalize flag.

CP02 Transmit data from normalize unit to Xd register.
Transmit data from normalize unit to Be register.
Clear Xd reservation flag.
Clear Be reservation flag.

Special situations

Special case operand

The normalize unit makes a special case test on (Xk) at the beginning of execution for this instruction. If one of these special cases is present the operand is copied unaltered to the Xi register. The shift count entered in the Bj register is zero for these cases. There are no condition flags set in the PSD register by the normalize unit. The special case formats are listed below and consist of partial overflow, complete overflow, and indefinite forms.

Special case formats:
(Xk) = 3777 xxxx xxxx xxxx xxxx
(Xk) = 4000 xxxx xxxx xxxx xxxx
(Xk) = 1777 xxxx xxxx xxxx xxxx
(Xk) = 6000 xxxx xxxx xxxx xxxx

Complete underflow

A complete underflow occurs for this instruction whenever (Xk) is not a special case and the normalizing process results in an unpacked exponent more negative than -1777 octal. In this situation a zero word is delivered to the Xi register. The sign of the operand is preserved in this process and (Xi) is either all zero bits, or all one bits, depending on the sign of the original operand. The shift count delivered to the Bj register is a result of considering the coefficient field of (Xk) without regard to the exponent. This quantity is therefore the value which would be appropriate for normalizing the operand if the exponent were in range. There are no condition flags set in the PSD register by the normalize unit.

Partial underflow

A partial underflow occurs for this instruction whenever (Xk) is not a special case and the normalizing process results in an unpacked exponent exactly equal to -1777 octal. In this situation the result is delivered to the Xi register and the Bj register as for a normal case even though subsequent computation may detect this operand as an underflow case.

Coefficient is zero

A zero coefficient in the operand for this instruction becomes nonzero with the addition of the round bit. In this case the round bit is shifted to the left by 48 bit positions in the normalizing process to become the most significant bit of the result coefficient. The shift count delivered to the Bj register is 48 decimal for this case. This case is superseded by one of the first two special situations described above if the operand is in a special case format, or if a complete underflow occurs.

Underflow operand

A special situation exists for executing this instruction if (Xk) is in one of the two formats listed below.

Underflow formats: (Xk) = 0000 xxxx xxxx xxxx xxxx
(Xk) = 7777 xxxx xxxx xxxx xxxx

These formats include positive and negative numbers in either a partial underflow or a complete underflow form. These cases are generally covered by one of the other special situations listed above. If (Xk) is a partial underflow quantity and the coefficient is normalized, the execution of this instruction will proceed as for a normal operand and the result will be an unaltered copy of the original operand. The shift count delivered to the Bj register will be zero. If (Xk) is a partial underflow quantity and the coefficient is not normalized, a complete underflow will occur as described under that heading.

If (Xk) is a complete underflow quantity the execution of this instruction will be dominated by the fact that the coefficient is zero. The round bit will be added and the shift count will be 48 decimal. This will cause a complete underflow and the result delivered to the Xi register will be the same as the original operand.

26ijk Unpack X to X, B

This instruction causes the boolean unit to read one operand from the Xk register, unpack this word from floating point format, and then deliver the coefficient to the Xi register and the exponent to the Bj register. The 60 bit word delivered to the Xi register consists of the lowest 48 bits unaltered from the original operand plus the upper 12 bits each equal to the original sign bit. This is a signed integer equal to the value of the coefficient in the original operand.

The 18 bit quantity delivered to the Bj register is a signed integer equal to the value of the exponent in the original operand. The 11 bit exponent field in the operand is altered to remove the bias and then sign extended to fill out the 18 bit quantity. The sign of the coefficient is removed in this process.

Four sample sets of operands and unpacked results are listed below in octal notation to illustrate the operation performed. These examples contain the four combinations of coefficient sign and exponent sign.

Sample operands:
```
(Xk) = 2034 4500 3333 2000 0077
(Xi) = 0000 4500 3333 2000 0077
(Bj) = 00 0034

(Xk) = 1743 4500 3333 2000 0077
(Xi) = 0000 4500 3333 2000 0077
(Bj) = 77 7743

(Xk) = 5743 3277 4444 5777 7700
(Xi) = 7777 3277 4444 5777 7700
(Bj) = 00 0034

(Xk) = 6034 3277 4444 5777 7700
(Xi) = 7777 3277 4444 5777 7700
(Bj) = 77 7743
```

This instruction is intended for converting a number from floating point format to fixed point format as quickly as possible. This process is the reciprocal of the process used to implement the 27 instruction.

Issue conditions

Xi register is free.
Xk register is free.
Bj register is free.
X register input path will be free in next clock period.
B register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The results will be delivered to the Xi register and the Bj register one clock period after the instruction issues. The Xi register and the Bj register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 26 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xk) to the boolean unit.
Set Xi reservation flag.
Set Bj reservation flag.
Set go boolean flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from boolean unit to Xd register.
Transmit data from boolean unit to Be register.
Clear Xd reservation flag.
Clear Be reservation flag.
Clear go boolean flag.

Special situations

There are no special case tests made in the execution of this instruction. There are no condition flags set in the PSD register by the boolean unit. The special operand formats are treated in the same manner as a normal operand. Various combinations of special floating point quantities are listed in octal notation in the examples below to indicate the unpacked results.

Special operands:

```
(Xk) = 3777 0000 0000 0000 0000
(Xi) = 0000 0000 0000 0000 0000
(Bj) = 00 1777

(Xk) = 4000 7777 7777 7777 7777
(Xi) = 7777 7777 7777 7777 7777
(Bj) = 00 1777

(Xk) = 1777 0000 0000 0000 0000
(Xi) = 0000 0000 0000 0000 0000
(Bj) = 77 7777

(Xk) = 6000 7777 7777 7777 7777
(Xi) = 7777 7777 7777 7777 7777
(Bj) = 77 7777

(Xk) = 0000 0000 0000 0000 0000
(Xi) = 0000 0000 0000 0000 0000
(Bj) = 77 6000

(Xk) = 7777 7777 7777 7777 7777
(Xi) = 7777 7777 7777 7777 7777
(Bj) = 77 6000
```

27ijk Pack X, B to X

This instruction causes the boolean unit to read (Xk) and (Bj), pack them into a single word in floating point format, and deliver this result to the Xi register. The coefficient for (Xi) is obtained from (Xk) treated as a signed integer. The exponent for (Xi) is obtained from (Bj) treated as a signed integer.

The lowest order 48 bits of (Xi) are copied directly from the lowest order 48 bits of (Xk). The sign bit in (Xi) is copied directly from the sign bit in (Xk). The exponent field in (Xi) is derived from (Bj) by extracting the lowest order 11 bits of (Bj) and modifying this quantity for exponent bias and coefficient sign.

Four sample sets of operands and packed results are listed below in octal notation to illustrate the operation performed. These examples contain the four combinations of coefficient sign and exponent sign.

Sample operands:

```
(Xk) = 0000 4500 3333 2000 0077
(Bj) = 00 0034
(Xi) = 2034 4500 3333 2000 0077

(Xk) = 0000 4500 3333 2000 0077
(Bj) = 77 7743
(Xi) = 1743 4500 3333 2000 0077

(Xk) = 7777 3277 4444 5777
(Bj) = 00 0034
(Xi) = 5743 3277 4444 5777

(Xk) = 7777 3277 4444 5777 7700
(Bj) = 77 7743
(Xi) = 6034 3277 4444 5777 7700
```

This instruction is intended for converting a number in fixed point format to floating point format as quickly as possible. This process is the reciprocal of the process used to implement the 26 instruction.

Issue conditions

Xi register is free.
Xk register is free.
Bj register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 27 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xk) to the boolean unit.
Transmit (Bj) to the boolean unit.
Set Xi reservation flag.
Set go boolean flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from boolean unit to the Xd register.
Clear Xd reservation flag.
Clear go boolean flag.

Special situations

(Xk) has magnitude greater than 48 bits

If (Xk) has more than 48 bits of significance the higher order bits will be ignored in packing this quantity into the floating point format. The lowest order 48 bits of (Xk) are masked out of the 60 bit word for the coefficient in (Xi). The sign bit in (Xk) is copied into (Xi) for the sign of the coefficient. The remaining bits of (Xk) are ignored.

(Bj) has magnitude greater than 10 bits

If (Bj) has more than 10 bits of significance an erroneous exponent will be packed into floating point format for (Xi). In this case the lowest order 11 bits of (Bj) will be masked out of the 18 bit quantity. The highest order of these 11 bits will be interpreted as the sign bit for the exponent. There are no error condition flags set in the PSD register by the boolean unit in this situation.

Designator j is zero

The j designator may be set to zero in this instruction to pack a fixed point integer into floating point format without using one of the active B registers.

Packing an indefinite quantity

If the lowest order 11 bits of (Bj) all have a value of one, an indefinite quantity will result in the floating point format. This will be the case if (Bj) is a negative zero quantity. There are no error condition flags set in the PSD register by the boolean unit in this situation.

Packing an overflow quantity

An overflow quantity will be generated in floating point format if (Bj) = 00 1777 octal. There are no error condition flags set in the PSD register by the boolean unit in this situation.

Packing an underflow quantity

An underflow quantity will be generated in floating point format if (Bj) = 77 6000 octal. There are no error condition flags set in the PSD register by the boolean unit in this situation.

30ijk Floating sum

This instruction causes the floating point add unit to read operands from two X registers, operate upon them to form a floating point sum, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). These operands are assumed to be numbers in floating point format. They may, or may not, be normalized. The result of the floating point add operation is delivered to the Xi register in floating point format. This result is not necessarily normalized.

The operands are not rounded in this operation. The two operands are unpacked from floating point format and the exponents compared. The unpacked coefficients are then positioned in a 99 bit ones complement adder so as to align bits of corresponding significance. A double precision ones complement sum is formed. A 48 bit result coefficient is then read from the upper half of this sum. If an overflow of the highest order coefficient bit occurred during the addition process the result coefficient is read from an alternate adder output path with a one bit displacement to include this overflow bit. The result exponent is corrected by one count in this case.

If the two operands have unlike signs the result coefficient may have leading zeros. There is no normalize operation built into this instruction to correct this situation. A separate normalize instruction must be programed if the result is to be kept in a normalized form.

This instruction is intended for use in floating point calculations where rounding of operands is not desired. This is the case in multiple precision arithmetic and in calculations involving error analysis.

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free three clock periods hence.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The execution time is a constant for all cases of operands. The result will be delivered to the Xi register three clock periods after the instruction issues. The Xi register will be reserved for the three clock periods from issue to delivery of data. The command timing for this instruction is listed below.

CP00 30 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register
Transmit (Xk) to the floating add unit.
Transmit (Xj) to the floating add unit.
Set Xi reservation flag.
Set go floating add flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Compare exponents.
Transmit coefficients from input register to shift register.
Clear go floating add flag.

CP02 Shift coefficients for alignment.
Transmit coefficients from shift register to adder.

CP03 Form double precision sum.
Transmit result from floating add unit to Xd register.
Clear Xd reservation flag.

Special situations

Clean miss

If the exponents of the two operands differ by more than 48 decimal the coefficient of the operand with the smaller exponent will be shifted off the end of the double precision adder. If the exponent difference is exactly 48 decimal the two coefficients will be aligned in a 96 bit field in the double precision adder with no bits matched in the add operation. In either of these cases the result of the floating add operation will be a copy of the operand with the larger exponent.

Result coefficient is zero

If the two operands are of equal magnitude and opposite sign the resulting sum will have a zero coefficient. The exponent delivered to the Xi register will be the same as the exponent for the operands even though the coefficient is zero. The sign of the result will be positive. No error condition flags will be set in the PSD register for this case.

Partial overflow

If the two operands are both in floating point range and one operand is at the upper limit of the floating point range, the resulting sum may overflow. In this case the resulting exponent will indicate the overflow condition, but the coefficient will be processed in a normal manner and the resulting floating point number will in fact be a correct representation of the sum. No error indication is made for this case, and no condition flags will be set in the PSD register. Subsequent use of this number as an operand in a floating point unit will, however, result in overflow detection.

One operand indefinite

If either operand is indefinite, or if both operands are indefinite, the result is indefinite. The operand coefficients are ignored in this case, and the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

Both operands overflow with different signs

If both operands have overflow exponents and the operand coefficients have different signs, the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

One operand overflow

If either operand has an overflow exponent, or if both operands have an overflow exponent and the coefficient signs agree, the result is an overflow word. In this case the operand coefficients are ignored, and the word delivered to the Xi register is a complete overflow with a zero coefficient. The sign of the resulting word is the same as the sign of the operand with the overflow exponent. The overflow condition flag is set in the PSD register for this case.

Underflow operand

An operand with an underflow exponent is treated as a normal operand in this instruction. No condition flags are set in the PSD register for this case. If both operands are zero and one operand is positive, the result will be a positive zero word. If both operands are negative zero words the result will be a negative zero word.

3ijk Floating difference

This instruction causes the floating point add unit to read operands from two X registers, operate upon them to form a floating point difference, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). These operands are assumed to be numbers in floating point format. They may, or may not, be normalized. The result of the floating point subtraction (Xj) minus (Xk) is delivered to the Xi register in floating point format. This result is not necessarily normalized.

The operands are not rounded in this operation. The two operands are unpacked from floating point format and the exponents compared. The unpacked coefficients are then positioned in a 99 bit ones complement adder so as to align bits of corresponding significance. A double precision ones complement difference is formed. A 48 bit result coefficient is then read from the upper half of this difference. If an overflow of the highest order coefficient bit occurred during the subtraction process the result coefficient is read from an alternate adder output path with a one bit displacement to include this overflow bit. The result exponent is corrected by one count in this case.

If the two operands have like signs the result coefficient may have leading zeros. There is no normalize operation built into this instruction to correct this situation. A separate normalize instruction must be programed if the result is to be kept in a normalized form.

This instruction is intended for use in floating point calculations where rounding of operands is not desired. This is the case in multiple precision arithmetic and in calculations involving error analysis.

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free three clock periods hence.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The execution time is a constant for all cases of operands. The result will be delivered to the Xi register three clock periods after the instruction issues. The Xi register will be reserved for the three clock periods from issue to delivery of data. The command timing for this instruction is listed below.

```
      31 instruction in the upper parcel of the CIW register.
      Instruction issues.
      Transmit the next instruction to upper parcel of CIW register.
      Transmit (Xk) to the floating add unit.
      Transmit (Xj) to the floating add unit.
      Set Xi reservation flag.
      Set go floating add flag.

CP01  Next instruction in the upper parcel of the CIW register.
      Instruction may issue.
      Compare exponents.
      Transmit coefficients from input register to shift register.
      Clear go floating add flag.

CP02  Shift coefficients for alignment.
      Transmit coefficients from shift register to adder.

      Form double precision difference.
      Transmit result from floating add unit to Xd register.
      Clear Xd reservation flag.
```

Special situations

Clean miss

If the exponents of the two operands differ by more than 48 decimal the coefficient of the operand with the smaller exponent will be shifted off the end of the double precision adder. If the exponent difference is exactly 48 decimal the two coefficients will be aligned in a 96 bit field in the double precision adder with no bits matched in the subtract operation. In either of these cases the result of the floating subtract operation will be a copy of the operand with the larger exponent.

Result coefficient is zero

If the two operands are identical the resulting difference will have a zero coefficient. The exponent delivered to the Xi register will be the same as the exponent for the operands even though the coefficient is zero. The sign of the result will be positive. No error condition flags will be set in the PSD register for this case.

Partial overflow

If the two operands are both in floating point range and one operand is at the upper limit of the floating point range, the resulting difference may overflow. In this case the resulting exponent will indicate the overflow condition, but the coefficient will be processed in a normal manner and the resulting floating point number will in fact be a correct representation of the difference. No error indication is made for this case, and no condition flags will be set in the PSD register. Subsequent use of this number as an operand in a floating point unit will, however, result in overflow detection.

One operand indefinite

If either operand is indefinite, or if both operands are indefinite, the result is indefinite. The operand coefficients are ignored in this case, and the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

Both operands overflow with same sign

If both operands have overflow exponents and the operand coefficients have the same sign, the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

Both operands overflow with different signs

If both operands have overflow exponents and the coefficient signs differ, the result is an overflow word. In this case the operand coefficients are ignored, and the word delivered to the Xi register is a complete overflow with a zero coefficient. The sign of the resulting word is the same as the sign of (Xj). The overflow condition flag is set in the PSD register for this case.

One operand overflow

If either operand has an overflow exponent and the other operand is in floating point range, or has an underflow exponent, the result is an overflow word. In this case the operand coefficients are ignored, and the word delivered to the Xi register is a complete overflow with a zero coefficient. The sign of the resulting word is the same as the sign of the operand with the overflow exponent. The overflow condition flag is set in the PSD register for this case.

Underflow operand

An operand with an underflow exponent is treated as a normal operand in this instruction. No condition flags are set in the PSD register for this case. If (Xj) is a negative zero word and (Xk) is a positive zero word, the result will be a negative zero word. The other three cases of both operands zero words will result in a positive zero word.

32ijk Floating double precision sum

This instruction causes the floating point add unit to read operands from two X registers, operate upon them to form a double precision floating point sum, and deliver the lower half of this result to a third X register. The operands for this instruction are (Xj) and (Xk). These operands are assumed to be numbers in floating point format. They may, or may not, be normalized. The result of the double precision add operation is delivered to the Xi register in floating point format. This result is not necessarily normalized.

The operands are not rounded in this operation. The two operands are unpacked from floating point format and the exponents compared. The unpacked coefficients are then positioned in a 99 bit ones complement adder so as to align bits of corresponding significance. A double precision ones complement sum is formed. A 48 bit result coefficient is then read from the lower half of this sum. The result exponent is exactly 48 decimal less than the exponent which would be delivered with the upper half of the double precision sum. If an overflow of the highest order coefficient bit occurred during the addition process the result coefficient is read from an alternate adder output path with a one bit displacement to take into account the overflow bit. The result exponent is corrected by one count in this case.

If the two operands have unlike signs the double precision sum may have leading zeros. There is no normalize operation built into this instruction to correct this situation. Whether this situation exists or not, there may be leading zeros in the lower half of the double precision sum. These zero bits are not detected, and the coefficient in the result for this instruction may have leading zeros.

This instruction is intended for use in floating point calculations involving double precision or multiple precision. This instruction together with the 30 instruction forms a double precision sum in two X registers with no loss of significance.

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free three clock periods hence
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The execution time is a constant for all cases of operands. The result will be delivered to the Xi register three clock periods after the instruction issues. The Xi register will be reserved for the three clock periods from issue to delivery of data. The command timing for this instruction is listed below.

32 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xk) to the floating add unit.
Transmit (Xj) to the floating add unit.
Set Xi reservation flag.
Set go floating add flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Compare exponents.
Transmit coefficients from input register to shift register.
Clear go floating add flag.

CP02 Shift coefficients for alignment.
Transmit coefficients from shift register to adder.

CP03 Form double precision sum.
Transmit result from floating add unit to Xd register.
Clear Xd reservation flag.

Special situations

Clean miss

If the exponents of the two operands differ by more than 48 decimal the coefficient of the operand with the smaller exponent will be shifted off the end of the double precision adder. If the exponent difference is exactly 48 decimal the two coefficients will be aligned in a 96 bit field with no bits matched in the add operation. In either of these cases the result delivered to the Xi register will contain a coefficient from the 48 bit field in the double precision adder corresponding to the lower half of a 96 bit sum. The exponent delivered to the Xi register will be exactly 48 decimal less than the exponent which would be delivered with the upper half of the 96 bit sum. If the difference of the operand exponents is greater than 48 decimal the lower half of this 96 bit sum will have leading zeros. If the difference of the operand exponents is 96 decimal or greater, the lower half of the 96 bit sum will be all zeros.

Coefficient sum is zero

If the two operands are of equal magnitude and opposite sign the resulting double precision coefficient sum will be zero. This condition is not sensed as a special case, and the exponent field in the result delivered to the Xi register will be the same value as for a nonzero coefficient. The sign of the resulting zero coefficient will be positive in this case. No error condition flags will be set in the PSD register for this case.

Partial overflow

If the two operands are in floating point range and one operand is at the upper limit of the floating point range, the resulting double precision sum may overflow and cause the exponent for the upper half to go out of range. This condition is not sensed in this instruction The exponent for the lower half of the double precision sum is 48 decimal less than this overflow value. The result delivered to the Xi register in this case is processed as a normal floating point result and no error condition flags are set in the PSD register.

Partial underflow

If the two operands are near the lower limit of the floating point range the exponent for the lower half of the double precision sum may be exactly -1777 octal. This result is processed as a normal floating point number, and no error condition flags are set in the PSD register. The resulting coefficient may be nonzero even though the exponent in the resulting word indicates an underflow condition. Subsequent use of this number as an operand in a floating point unit may, however, result in underflow detection.

Complete underflow

If the two operands are near the lower limit of the floating point range the exponent for the lower half of the double precision sum may be less than -1777 octal. This condition is sensed as a special case, and the result delivered to the Xi register is a complete underflow word with a zero coefficient. The sign of the result will be the same as the sign of the operand with the larger exponent. If the two operands have identical exponents the sign of the result will be the same as the sign of (Xk). The underflow condition flag will be set in the PSD register for this case. The special case test is made before the coefficients are added. The result of this addition is ignored in this case, and an overflow of the highest order coefficient bit will not bring the result back into range.

Underflow operand

An operand with an underflow exponent is treated as a normal operand in this instruction. No condition flags are set in the PSD register if the other operand is sufficiently large so that the result does not underflow the floating point range. If the other operand is near the lower limit of the floating point range the result may be either a partial underflow, or a complete underflow, as described above.

One operand indefinite

If either operand is indefinite, or if both operands are indefinite, the result is indefinite. The operand coefficients are ignored in this case, and the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

Both operands overflow with different signs

If both operands have overflow exponents and the operand coefficients have different signs, the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

One operand overflow

If either operand has an overflow exponent, or if both operands have an overflow exponent and the coefficient signs agree, the result is an overflow word. In this case the operand coefficients are ignored, and the word delivered to the Xi register is a complete overflow with a zero coefficient. The sign of the resulting word is the same as the sign of the operand with the overflow exponent. The overflow condition flag is set in the PSD register for this case.

33ijk Floating double precision difference

This instruction causes the floating point add unit to read operands from two X registers, operate upon them to form a double precision floating point difference, and deliver the lower half of this result to a third X register. The operands for this instruction are (Xj) and (Xk). These operands are assumed to be numbers in floating point format. They may, or may not, be normalized. The result of the double precision subtraction (Xj) minus (Xk) is delivered to the Xi register in floating point format. This result is not necessarily normalized.

The operands are not rounded in this operation. The two operands are unpacked from floating point format and the exponents compared. The unpacked coefficients are then positioned in a 99 bit ones complement adder so as to align bits of corresponding significance. A double precision ones complement difference is formed. A 48 bit result coefficient is then read from the lower half of this difference. The result exponent is exactly 48 decimal less than the exponent which would be delivered with the upper half of the double precision difference. If an overflow of the highest order coefficient bit occurred during the subtraction process the result coefficient is read from an alternate adder output path with a one bit displacement to take into account the overflow bit. The result exponent is corrected by one count in this case.

If the two operands have like signs the double precision difference may have leading zeros. There is no normalize operation built into this instruction to correct this situation. Whether this situation exists or not, there may be leading zeros in the lower half of the double precision difference. These zero bits are not detected, and the coefficient in the result for this instruction may have leading zeros.

This instruction is intended for use in floating point calculations involving double precision or multiple precision. This instruction together with the 31 instruction forms a double precision difference in two X registers with no loss of significance.

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free three clock periods hence.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The execution time is a constant for all cases of operands. The result will be delivered to the Xi register three clock periods after the instruction issues. The Xi register will be reserved for the three clock periods from issue to delivery of data. The command timing for this instruction is listed below.

CP00 33 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xk) to the floating add unit.
Transmit (Xj) to the floating add unit.
Set Xi reservation flag.
Set go floating add flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Compare exponents.
Transmit coefficients from input register to shift register.
Clear go floating add flag.

CP02 Shift coefficients for alignment.
Transmit coefficients from shift register to adder.

CP03 Form double precision difference.
Transmit result from floating add unit to Xd register.
Clear Xd reservation flag.

Special situations

Clean miss

If the exponents of the two operands differ by more than 48 decimal the coefficient of the operand with the smaller exponent will be shifted off the end of the double precision adder. If the exponent difference is exactly 48 decimal the two coefficients will be aligned in a 96 bit field with no bits matched in the subtract operation. In either of these cases the result delivered to the Xi register will contain a coefficient from the 48 bit field in the double precision adder corresponding to the lower half of a 96 bit difference. The exponent delivered to the Xi register will be exactly 48 decimal less than the exponent which would be delivered with the upper half of the 96 bit difference. If the difference of the operand exponents is greater than 48 decimal the lower half of this 96 bit difference will have leading zeros. If the difference of the operand exponents is 96 decimal or greater, the lower half of the 96 bit difference will be all zeros.

Coefficient sum is zero

If the two operands are identical the resulting double precision coefficient difference will be zero. This condition is not sensed as a special case, and the exponent field in the result delivered to the Xi register will be the same value as for a nonzero coefficient. The sign of the resulting zero coefficient will be positive in this case. No error condition flags will be set in the PSD register for this case.

Partial overflow

If the two operands are in floating point range and one operand is at the upper limit of the floating point range, the resulting double precision difference may overflow and cause the exponent for the upper half to go out of range. This condition is not sensed in this instruction. The exponent for the lower half of the double precision difference is 48 decimal less than this overflow value. The result delivered to the Xi register in this case is processed as a normal floating point result and no error condition flags are set in the PSD register.

Partial underflow

If the two operands are near the lower limit of the floating point range the exponent for the lower half of the double precision difference may be exactly -1777 octal. This result is processed as a normal floating point number, and no error condition flags are set in the PSD register. The resulting coefficient may be nonzero even though the exponent in the resulting word indicates an underflow condition. Subsequent use of this number as an operand in a floating point unit may, however, result in underflow detection.

Complete underflow

If the two operands are near the lower limit of the floating point range the exponent for the lower half of the double precision difference may be less than -1777 octal. This condition is sensed as a special case, and the result delivered to the Xi register is a complete underflow word with a zero coefficient. The sign of the result will be the same as the sign of (Xj) if (Xj) has the larger exponent. The sign of the result will be the complement of the sign of (Xk) if (Xk) has the larger exponent, or if the exponents are equal. The underflow condition flag will be set in the PSD register for this case. The special case test is made before the coefficients are subtracted. The result of this subtraction is ignored in this case, and an overflow of the highest order coefficient bit will not bring the result back into range.

Underflow operand

An operand with an underflow exponent is treated as a normal operand in this instruction. No condition flags are set in the PSD register if the other operand is sufficiently large so that the result does not underflow the floating point range. If the other operand is near the lower limit of the floating point range the result may be either a partial underflow, or a complete underflow, as described above.

One operand indefinite

If either operand is indefinite, or if both operands are indefinite, the result is indefinite. The operand coefficients are ignored in this case, and the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

Both operands overflow with similar signs

If both operands have overflow exponents and the operand coefficients have similar signs, the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

One operand overflow

If either operand has an overflow exponent, or if both operands have an overflow exponent and the coefficient signs disagree, the result is an overflow word. In this case the operand coefficients are ignored, and the word delivered to the Xi register is a complete overflow with a zero coefficient. The sign of the resulting word is the same as the sign of (Xj) if (Xj) has the overflow exponent. The sign of the resulting word is the complement of the sign of (Xk) if (Xk) has the overflow exponent. The overflow condition flag is set in the PSD register for this case.

34ijk Round floating sum

This instruction causes the floating point add unit to read operands from two X registers, operate upon them to form a rounded floating point sum, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). These operands are assumed to be numbers in floating point format. They may, or may not, be normalized. The result of the floating point add operation is delivered to the Xi register in floating point format. This result is not necessarily normalized.

The floating point add unit unpacks the two operands from floating point format and compares the exponents. The unpacked coefficients are then positioned in a 99 bit ones complement adder so as to align bits of corresponding significance. The two coefficients are rounded according to the rules described below. A double precision ones complement sum is formed. A 48 bit result coefficient is then read from the upper half of this sum. If an overflow of the highest order coefficient bit occurred during the addition process the result coefficient is read from an alternate adder output path with a one bit displacement to include this overflow bit. The result exponent is corrected by one count in this case.

If the two operands have unlike signs the result coefficient may have leading zeros. There is no normalize operation built into this instruction to correct this situation. A separate normalize instruction must be programed if the result is to be kept in a normalized form.

This instruction is intended for use in floating point calculations involving single precision accuracy. For multiple precision calculations the 30 instruction and 32 instruction must be used.

Rounding

Rounding of the operand coefficients occurs just prior to the double precision add operation. At this time the two 48 bit coefficients are positioned in the 99 bit ones complement adder with an offset corresponding to the difference of the exponents. A round bit is always added to the coefficient corresponding to the larger exponent. If the exponents are equal the round bit is added to the coefficient for (Xk). The round bit is equal to the complement of the sign bit and is inserted immediately to the right of the lowest order bit in the coefficient. This has the effect of increasing the magnitude

of the coefficient by one half of the least significant bit. A second round bit is added in a corresponding manner to the other coefficient if both operands were normalized, or if the operands had unlike signs.

The amount of error introduced by the rounding operation is a function of the relative magnitudes of the operands. If the two operands differ significantly in the exponent field the rounding is relatively free of bias and the maximum error is bounded by $+\frac{1}{2}$ and $-\frac{1}{2}$ of the least significant bit of the larger coefficient. If the operands differ by only a few counts in the exponent field the rounding introduces some bias because of the discrete combinations involved. An additional complication is introduced by the possibility of overflow during the additional process. If an overflow occurs the result coefficient is truncated one bit position higher in the double precision sum. This introduces a negative bias on the rounding operation whenever it occurs.

Three tables are presented on the following page to indicate the rounding error for various combinations of operand values. The first of these tables considers the case of both operands normalized and the signs of the operands alike. It assumes a random distribution of bits at the lower order end of the coefficients in determining the round bias. It also assumes a random distribution of bits near the upper end of the coefficients in determining the probability of overflow. The dissymmetry of the maximum round error in this table is due to the overflow effects on the rounding position.

The second table presented on the following page is intended to indicate the rounding errors for the case of non-normalized operands. This table is not quite correct for this case because there is some probability of overflow even with one operand not normalized. The rounding error for operands that are nearly normalized will fall somewhere between the results of the first and second tables.

The third table presented on the following page is for the case of floating addition with the operand signs different. There is no possibility of overflow during the addition process for this case. The results are the same for normalized or non-normalized operands.

Rounding error tables

EXD = Exponent difference.
POV = Probability of overflow (in octal).
NOB = Average round bias for no overflow cases (in octal).
OVB = Average round bias for overflow cases (in octal).
AVB = Average round bias considering probability of overflow (in octal).
LPE = Largest positive round error for all cases (in octal).
LNE = Largest negative round error for all cases (in octal).

Floating add Signs alike Both operands normalized

EXD	POV	NOB	OVB	AVB	LPE	LNE
0	1.00	0.00	+.20	+.20	+.40	-.40
1	.60	+.20	-.10	-.02	+.40	-.40
2	.30	+.10	-.14	+.004	+.40	-.50
3	.14	+.04	-.16	+.005	+.40	-.54
4	.06	+.02	-.17	+.0032	+.40	-.56
5	.03	+.01	-.174	+.00164	+.40	-.57
large	.00	+.00	-.20	+.00	+.40	-.60

Floating add - Signs alike No overflow

EXD	NOB	LPE	LNE
0	0.00	0.00	0.00
1	+.20	+.40	0.00
2	+.10	+.40	-.20
3	+.04	+.40	-.30
4	+.02	+.40	-.34
5	+.01	+.40	-.36
large	0.00	+.40	-.40

Floating add Signs unlike

EXD	AVB	LPE	LNE
0	0.00	0.00	0.00
1	-.20	0.00	-.40
2	-.10	+.20	-.40
3	-.04	+.30	-.40
4	-.02	+.34	-.40
5	-.01	+.36	-.40
large	0.00	+.40	-.40

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free three clock periods hence.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The execution time is a constant for all cases of operands. The result will be delivered to the Xi register three clock periods after the instruction issues. The Xi register will be reserved for the three clock periods from issue to delivery of data. The command timing for this instruction is listed below.

34 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xk) to the floating add unit.
Transmit (Xj) to the floating add unit.
Set Xi reservation flag.
Set go floating add flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Compare exponents.
Transmit coefficients from input register to shift register.
Clear go floating add flag.

CP02 Shift coefficients for alignment.
Transmit coefficients from shift register to adder.

CP03 Form rounded sum.
Transmit result from floating add unit to Xd register.
Clear Xd reservation flag.

Special situations

Clean miss

If the exponents of the two operands differ by more than 48 decimal the coefficient of the operand with the smaller exponent will be shifted off the end of the double precision adder. In this case the result of the floating add operation will be a copy of the operand with the larger exponent. If the exponents of the two operands differ by exactly 48 decimal the two operand coefficients will be aligned in a 96 bit field in the double precision adder with no bits matched in the add operation. However, the round bit for the larger number will be aligned with the highest order bit for the smaller number. In this case the result of the floating add operation will be a rounded version of the operand with the larger exponent.

Result coefficient is zero

If the two operands are of equal magnitude and opposite sign the resulting sum will have a zero coefficient. The exponent delivered to the Xi register will be the same as the exponent for the operands even though the coefficient is zero. The sign of the result will be positive. No error condition flags will be set in the PSD register for this case.

Partial overflow

If the two operands are both in floating point range and one operand is at the upper limit of the floating point range, the resulting sum may overflow. In this case the resulting exponent will indicate the overflow condition, but the coefficient will be processed in a normal manner and the resulting floating point number will in fact be a correct representation of the sum. No error indication is made for this case, and no condition flags will be set in the PSD register. Subsequent use of this number as an operand in a floating point unit will, however, result in overflow detection.

One operand indefinite

If either operand is indefinite, or if both operands are indefinite, the result is indefinite. The operand coefficients are ignored in this case, and the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

Both operands overflow with different signs

If both operands have overflow exponents and the operand coefficients have different signs, the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

One operand overflow

If either operand has an overflow exponent, or if both operands have an overflow exponent and the coefficient signs agree, the result is an overflow word. In this case the operand coefficients are ignored, and the word delivered to the Xi register is a complete overflow with a zero coefficient. The sign of the resulting word is the same as the sign of the operand with the overflow exponent. The overflow condition flag is set in the PSD register for this case.

Underflow operand

An operand with an underflow exponent is treated as a normal operand in this instruction. No condition flags are set in the PSD register for this case. If both operands are zero, either positive zero, or negative zero, in any combination, the result will be a positive zero word.

35ijk Round floating difference

This instruction causes the floating point add unit to read operands from two X registers, operate upon them to form a rounded floating point difference, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). These operands are assumed to be numbers in floating point format. They may, or may not, be normalized. The result of the floating point subtraction (Xj) minus (Xk) is delivered to the Xi register in floating point format. This result is not necessarily normalized.

The floating point add unit unpacks the two operands from floating point format and compares the exponents. The unpacked coefficients are then positioned in a 99 bit ones complement adder so as to align bits of corresponding significance. The two coefficients are rounded according to the rules described below. A double precision ones complement difference is formed. A 48 bit result coefficient is then read from the upper half of this difference. If an overflow of the highest order coefficient bit occurred during the subtraction process the result coefficient is read from an alternate adder output path with a one bit displacement to include this overflow bit. The result exponent is corrected by one count in this case.

If the two operands have like signs the result coefficient may have leading zeros. There is no normalize operation built into this instruction to correct this situation. A separate normalize instruction must be programed if the result is to be kept in a normalized form.

This instruction is intended for use in floating point calculations involving single precision accuracy. For multiple precision calculations the 31 instruction and 33 instruction must be used.

Rounding

Rounding of the operand coefficients occurs just prior to the double precision subtract operation. At this time the two 48 bit coefficients are positioned in the 99 bit ones complement adder with an offset corresponding to the difference of the exponents. A round bit is always added to the coefficient corresponding to the larger exponent. If the exponents are equal the round bit is added to the coefficient for (Xk). The round bit is equal to the complement of the sign bit and is inserted immediately to the right of the lowest order bit in the coefficient. This has the effect of increasing the magnitude

of the coefficient by one half of the least significant bit. A second round bit is added in a corresponding manner to the other coefficient if both operands were normalized, or if the operands had like signs.

The amount of error introduced by the rounding operation is a function of the relative magnitudes of the operands. If the two operands differ significantly in the exponent field the rounding is relatively free of bias and the maximum error is bounded by $+\frac{1}{2}$ and $-\frac{1}{2}$ of the least significant bit of the larger coefficient. If the operands differ by only a few counts in the exponent field the rounding introduces some bias because of the discrete combinations involved. An additional complication is introduced by the possibility of overflow during the additional process. If an overflow occurs the result coefficient is truncated one bit position higher in the double precision difference. This introduces a negative bias on the rounding operation whenever it occurs.

Three tables are presented on the following page to indicate the rounding error for various combinations of operand values. The first of these tables considers the case of both operands normalized and the signs of the operands unlike. It assumes a random distribution of bits at the lower order end of the coefficients in determining the round bias. It also assumes a random distribution of bits near the upper end of the coefficients in determining the probability of overflow. The dissymmetry of the maximum round error in this table is due to the overflow effects on the rounding position.

The second table presented on the following page is intended to indicate the rounding errors for the case of non-normalized operands. This table is not quite correct for this case because there is some probability of overflow even with one operand not normalized. The rounding error for operands that are nearly normalized will fall somewhere between the results of the first and second tables.

The third table presented on the following page is for the case of floating subtraction with the operand signs alike. There is no possibility of overflow during the subtraction process for this case. The results are the same for normalized or non-normalized operands.

Rounding error tables

EXD = Exponent difference.
POV = Probability of overflow (in octal).
NOB = Average round bias for no overflow cases (in octal).
OVB = Average round bias for overflow cases (in octal).
AVB = Average round bias considering probability of overflow (in octal).
LPE = Largest positive round error for all cases (in octal).
LNE = Largest negative round error for all cases (in octal).

Floating subtract - Signs unlike - Both operands normalized

EXD	POV	NOB	OVB	AVB	LPE	LNE
0	1.00	0.00	+.20	+.20	+.40	-.40
1	.60	+.20	-.10	-.02	+.40	-.40
2	.30	+.10	-.14	+.004	+.40	-.50
3	.14	+.04	-.16	+.005	+.40	-.54
4	.06	+.02	-.17	+.0032	+.40	-.56
5	.03	+.01	-.174	+.00164	+.40	-.57
large	.00	+.00	-.20	+.00	+.40	-.60

Floating subtract - Signs unlike - No overflow

	NOB	LPE	LNE
0	0.00	0.00	0.00
1	+.20	+.40	0.00
2	+.10	+.40	-.20
3	+.04	+.40	-.30
4	+.02	+.40	-.34
5	+.01	+.40	-.36
large	0.00	+.40	-.40

Floating subtract - Signs alike

	AVB	LPE	LNE
0	0.00	0.00	0.00
1	-.20	0.00	-.40
2	-.10	+.20	-.40
3	-.04	+.30	-.40
4	-.02	+.34	-.40
5	-.01	+.36	-.40
large	0.00	+.40	-.40

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free three clock periods hence.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The execution time is a constant for all cases of operands. The result will be delivered to the Xi register three clock periods after the instruction issues. The Xi register will be reserved for the three clock periods from issue to delivery of data. The command timing for this instruction is listed below.

CP00 35 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xk) to the floating add unit.
Transmit (Xj) to the floating add unit.
Set Xi reservation flag.
Set go floating add flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Compare exponents.
Transmit coefficients from input register to shift register.
Clear go floating add flag.

CP02 Shift coefficients for alignment.
Transmit coefficients from shift register to adder.

CP03 Form rounded difference.
Transmit result from floating add unit to Xd register.
Clear Xd reservation flag.

Special situations

Clean miss

If the exponents of the two operands differ by more than 48 decimal the coefficient of the operand with the smaller exponent will be shifted off the end of the double precision adder. In this case the result of the floating subtract operation will be a copy of the operand with the larger exponent. If the exponents of the two operands differ by exactly 48 decimal the two operand coefficients will be aligned in a 96 bit field in the double precision adder with no bits matched in the add operation. However, the round bit for the larger number will be aligned with the highest order bit for the smaller number. In this case the result of the floating subtract operation will be a rounded version of the operand with the larger exponent.

Result coefficient is zero

If the two operands are identical the resulting difference will have a zero coefficient. The exponent delivered to the Xi register will be the same as the exponent for the operands even though the coefficient is zero. The sign of the result will be positive. No error condition flags will be set in the PSD register for this case.

Partial overflow

If the two operands are both in floating point range and one operand is at the upper limit of the floating point range, the resulting difference may overflow. In this case the resulting exponent will indicate the overflow condition, but the coefficient will be processed in a normal manner and the resulting floating point number will in fact be a correct representation of the difference. No error indication is made for this case, and no condition flags will be set in the PSD register. Subsequent use of this number as an operand in a floating point unit will, however, result in overflow detection.

One operand indefinite

If either operand is indefinite, or if both operands are indefinite, the result is indefinite. The operand coefficients are ignored in this case, and the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

Both operands overflow with same sign

If both operands have overflow exponents and the operand coefficients have the same sign, the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

One operand overflow

If either operand has an overflow exponent, or if both operands have an overflow exponent and the coefficient signs disagree, the result is an overflow word. In this case the operand coefficients are ignored, and the word delivered to the Xi register is a complete overflow with a zero coefficient. The sign of the resulting word is the same as the sign of (Xj) if (Xj) has the overflow exponent. The sign of the result is the complement of the sign of (Xk) if (Xk) has the overflow exponent. The overflow condition flag is set in the PSD register for this case.

Underflow operand

An operand with an underflow exponent is treated as a normal operand in this instruction. No condition flags are set in the PSD register for this case. If both operands are zero, either positive zero, or negative zero, in any combination, the result will be a positive zero word.

36ijk Integer sum

This instruction causes the long add unit to read operands from two X registers, operate upon them to form a 60 bit integer sum, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). These operands are assumed to be signed integers. The resulting integer sum is delivered to the Xi register.

The long add unit executes this instruction in a 60 bit ones complement mode. The two operands are read directly to a 60 bit integer adder. The resulting sum is delivered directly to the Xi register. There are no special cases sensed. No detection is made of overflow.

This instruction is intended for addition of integers too large for handling in the increment unit. This instruction is also useful in merging and comparing data fields during data processing.

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 36 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the long add unit.
Transmit (Xk) to the long add unit.
Set Xi reservation flag.
Set go long add flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit result from long add unit to Xd register.
Clear Xd reservation flag.
Clear go long add flag.

Special situations

Both operands zero

If both operands are zero the result is zero. If either operand is positive zero the result is positive zero. If both operands are negative zero the result is negative zero.

Designators j and k have the same value

If the j and k designators have the same value the designated 60 bit operand is added to itself and the resulting sum delivered to the Xi register.

Designators i and k have the same value
Designators i and j have the same value

If the i designator has the same value as the j designator, or the k designator, this instruction becomes a replace add instruction. The initial (Xi) is added to the other operand and the result then stored back in the Xi register.

7ijk	Intege difference

This instruction causes the long add unit to read operands from two X registers, operate upon them to form a 60 bit integer difference, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). These operands are assumed to be signed integers. The resulting integer difference (Xj) minus (Xk) is delivered to the Xi register.

The long add unit executes this instruction in a 60 bit ones complement mode. The two operands are read directly to a 60 bit integer adder. (Xj) is transmitted unaltered from the register to the adder. (Xk) is complemented in the transmission from the register to the adder. The resulting sum of (Xj) and the complement of (Xk) is delivered directly to the Xi register. There are no special cases sensed. No detection is made of overflow.

This instruction is intended for subtraction of integers too large for handling in the increment unit. This instruction is also useful in comparing data fields during data processing.

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
X register input path will be free in next clock period
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 37 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the long add unit.
Transmit (Xk) to the long add unit.
Set Xi reservation flag.
Set go long add flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit result from long add unit to Xd register.
Clear Xd reservation flag.
Clear go long add flag.

Special situations

Both operands zero

If (Xj) is a negative zero quantity, and (Xk) is a positive zero quantity, the result is a negative zero quantity. The other three combinations of positive and negative zero operands result in a positive zero quantity.

Designators j and k have the same value

If the j and k designators have the same value the designated 60 bit operand is subtracted from itself. The result is a positive zero result in the Xi register.

Designators i and j have the same value
Designators i and k have the same value

If the i designator has the same value as the j designator, or the k designator, this instruction becomes a replace subtract instruction. The initial (Xi) is read as an operand, and the resulting difference is then stored in the same register.

40ijk | Floating product

This instruction causes the multiply unit to read operands from two X registers, operate upon them to form a floating point product, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). These operands are assumed to be numbers in floating point format. They may, or may not, be normalized. The result of the floating point multiply operation is delivered to the Xi register in floating point format. If both operands were normalized the result will also be normalized. If both operands were not normalized the result will not be normalized.

The operands are not rounded in this operation. The two operands are unpacked from floating point format. The exponents are added with a correction factor to determine the exponent for the result. The coefficients are multiplied as signed integers to form a 96 bit integer product. The upper half of this product is then extracted to form the coefficient for the result. An alternate output path is provided with a one bit position displacement to normalize the result coefficient if the original operands were normalized and the double precision product has only 95 significant bits. The exponent for the result is corrected by one count in this case.

If the two operands are not both normalized the resulting double precision product will have less than 96 significant bits. No test is made for the position of the most significant bit in the product for this case. The upper 48 bits are read from the 96 bit positions in the double precision product register, and leading zeros will occur in the result coefficient. The alternate path is not used in this case even though the one bit displacement may have normalized the result.

This instruction is intended for use in floating point calculations where rounding of operands is not desired. This is the case in multiple precision arithmetic and in calculations involving error analysis.

Issue conditions

Xi register free.
Xj register free.
Xk register free.
Multiply unit free.
X register input path will be free four clock periods hence.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register four clock periods after the instruction issues. The Xi register will be reserved for the four clock periods from issue to delivery of data. The multiply unit will be free two clock periods after this instruction issues. The command timing for this instruction is listed below.

CP00 40 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the multiply unit.
Transmit (Xk) to the multiply unit.
Set Xi reservation flag.
Set multiply unit busy flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Form first 24 by 48 bit product.
Clear multiply unit busy flag.

CP02 Form second 24 by 48 bit product.

Merge two 24 by 48 bit products into 99 bit adder.

Transmit result from multiply unit to Xd register.
Clear Xd reservation flag.

Special situations

Result coefficient is zero

If the two operands are not both normalized the upper half of the double precision product may be all zeros. This situation is not sensed, and the exponent for the result will be processed without regard to the zero coefficient. This will result in a zero coefficient and a nonzero exponent. No error flags are set in the PSD register for this case.

Partial overflow

A partial overflow occurs for this instruction whenever the exponent computation results in exactly +1777 octal and the result coefficient is taken from a double precision product with 96 bits of significance. There are no error condition flags set in the PSD register for this case, and the result is delivered to the Xi register in a normal manner. Subsequent use of this result as an operand in a floating point unit will, however, result in overflow detection. If the exponent computation in this instruction results in exactly +1777 octal and the alternate output path is used, the exponent delivered to the Xi register will be reduced one count and the result will be in floating point range.

Complete overflow

A complete overflow occurs for this instruction whenever the exponent computation results in an exponent greater than +1777 octal. This situation is sensed as a special case, and a complete overflow word with proper sign, overflow exponent, and zero coefficient is delivered to the Xi register. The coefficient calculation is ignored for this case, and the overflow condition flag is set in the PSD register.

Partial underflow

A partial underflow occurs for this instruction whenever the exponent computation results in exactly -1776 octal and the alternate output path is used from the double precision adder to normalize the result coefficient. In this case the exponent delivered to the Xi register is reduced one count and creates an underflow exponent with a valid coefficient. There are no condition flags set in the PSD register for this case. Subsequent use of this result in a floating point unit may, however, result in underflow detection.

Complete underflow

A complete underflow occurs for this instruction whenever the exponent computation results in less than -1776 octal. This situation is sensed as a special case, and a complete zero word with proper sign is delivered to the Xi register. The coefficient calculation is ignored in this case, and the underflow condition flag is set in the PSD register.

One operand indefinite

If either operand is indefinite, or if both operands are indefinite, the result is indefinite. The operand coefficients are ignored in this case, and the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

One operand overflow

If one operand has an overflow exponent and the other operand is in floating point range, or if both operands have overflow exponents, the result is a complete overflow word delivered to the Xi register. The coefficients of the operands are ignored in this case, and the result has a zero coefficient. The sign of the result is calculated in the same manner as for operands in range. The overflow condition flag is set in the PSD register for this case.

One operand underflow

If one operand has an underflow exponent and the other operand is in floating point range, or if both operands have underflow exponents, the result is a complete underflow word delivered to the Xi register. The coefficients of the operands are ignored in this case, and the result has a zero coefficient. The sign of the result is calculated in the same manner as for operands in range. The underflow condition flag is set in the PSD register for this case.

Underflow times overflow

If one operand has an underflow exponent and the other operand has an overflow exponent the result is indefinite. The operand coefficients are ignored in this case, and the word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

41ijk	Round floating product

This instruction causes the multiply unit to read operands from two X registers, operate upon them to form a rounded floating point product, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). These operands are assumed to be numbers in floating point format. They may, or may not, be normalized. The result of the floating point multiply operation is delivered to the Xi register in floating point format. If both operands were normalized the result will also be normalized. If both operands were not normalized the result will not be normalized.

The multiply unit unpacks the two operands from floating point format. The exponents are added with a correction factor to determine the exponent for the result. The coefficients are multiplied as signed integers to form a 96 bit integer product. A rounding bit is added in bit position 46 of this product. The upper half of this product is then extracted to form the coefficient for the result. An alternate output path is provided with a one bit position displacement to normalize the result coefficient if the original operands were normalized and the double precision product has only 95 bits of significance. The exponent for the result is corrected by one count in this case.

If the two operands are not both normalized the resulting double precision product will have less than 96 significant bits. No test is made for the position of the most significant bit in the product for this case. The upper 48 bits are read from the 96 bit positions in the double precision product register in this case, and leading zeros will occur in the result coefficient. The alternate path is not used in this case even though the one bit displacement may have normalized the result.

This instruction is intended for use in single precision floating point calculations. For multiple precision calculations the 40 instruction and 42 instruction must be used.

Rounding

Rounding of the result coefficient occurs in the final addition of the partial products to form a 96 bit double precision result. The rounding is accomplished by adding a bit in position 46 of the adder This additional bit has the effect of reducing the maximum amount of truncation error and also reducing the average bias.

If both operands are normalized there are two cases to consider in the calculation of rounding error. One case, with a 60 per cent probability of occurring, results from reading the result coefficient from the upper 48 bit positions in the 96 bit product register. The other case, with a 40 per cent probability of occurring, results from reading the coefficient from the alternate output path with a one bit displacement to normalize the result. These two cases are summarized in the table below. The third table entry indicates the error for both operands normalized with random coefficient values. The fourth table entry indicates the error for both operands not normalized.

LPE - Largest positive round error (decimal).
LNE - Largest negative round error (decimal).
AVB - Average bias for random coefficients (decimal).

	LPE	LNE	AVB
Normal path (60%)	+.25	-.75	-.25
Alternate path (40%)	+.50	-.50	0.0
Random normalized	+.50	-.75	-.15
Random not normalized	+.25	-.75	-.25

Issue conditions

Xi register free.
Xj register free.
Xk register free.
Multiply unit free.
X register input path will be free four clock periods hence.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register four clock periods after the instruction issues. The Xi register will be reserved for the four clock periods from issue to delivery of data. The multiply unit will be free two clock periods after this instruction issues. The command timing for this instruction is listed below.

CP00 41 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the multiply unit.
Transmit (Xk) to the multiply unit.
Set Xi reservation flag.
Set multiply unit busy flag.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Form first 24 by 48 bit product.
Clear multiply unit busy flag.

Form second 24 by 48 bit product.

Merge two 24 by 48 bit products into 99 bit adder.

CP04 Transmit result from multiply unit to Xd register.
Clear Xd reservation flag.

Special situations

The special situations for this instruction are the same as the special situations for the 40 instruction.

42ijk Floating double precision product

This instruction causes the multiply unit to read operands from two X registers, operate upon them to form a floating point double precision product, and deliver the lower half of this result to a third X register. The operands for this instruction are (Xj) and (Xk). These operands are assumed to be numbers in floating point format. They may, or may not, be normalized. The lower half of the double precision product is delivered to the Xi register in floating point format. This result is not necessarily normalized.

The operands are not rounded in this operation. The two operands are unpacked from floating point format. The exponents are added to determine the exponent for the result. The result exponent for this instruction is exactly 48 less than the exponent calculation for a 40 instruction. The coefficients are multiplied as signed integers to form a 96 bit integer product. The lower half of this product is then extracted to form the coefficient for the result. An alternate output path is provided with a one bit position displacement to correspond with the normalizing alternate path for the 40 instruction when both operands are normalized and the double precision product has only 95 significant bits. The exponent for the result is corrected by one count if this path is used.

If the two operands are not both normalized the resulting double precision product will have less than 96 significant bits. The alternate output path will never be used in this case. No test is made for the position of the most significant bit in the product. The lower 48 bits are always read from the 96 bit product register in this case.

This instruction is intended for use in multiple precision floating point calculations. This instruction is also intended for integer multiplication where the operands have non-normalized integer coefficients with less than 24 significant bits.

Issue conditions

Xi register free.
Xj register free.
Xk register free.
Multiply unit free.
X register input path will be free four clock periods hence.
No SAS backup condition.

Execution Time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register four clock periods after the instruction issues. The Xi register will be reserved for the four clock periods from issue to delivery of data. The multiply unit will be free two clock periods after this instruction issues. The command timing for this instruction is listed below.

```
CP00  42 instruction in the upper parcel of the CIW register.
      Instruction issues.
      Transmit the next instruction to upper parcel of CIW register
      Transmit (Xj) to the multiply unit.
      Transmit (Xk) to the multiply unit.
      Set Xi reservation flag.
      Set multiply unit busy flag.

      Next instruction in the upper parcel of the CIW register.
      Instruction may issue.
      Form first 24 by 48 bit product.
      Clear multiply unit busy flag.

      Form second 24 by 48 bit product.

      Merge two 24 by 48 bit products into 99 bit adder.

      Transmit result from multiply unit to Xd register.
      Clear Xd reservation flag.
```

Special situations

Integer product

This instruction may be used to form the product of two integers providing the resulting product will not exceed 48 bits of significance. The operands must be packed into floating point format (generally by using register B0) before executing this instruction. The result must be unpacked to obtain the integer product.

Partial overflow

A partial overflow occurs for this instruction whenever the exponent computation results in exactly +1777 octal and the result coefficient is taken from a double precision product with 96 bits of significance. There are no error condition flags set in the PSD register for this case, and the result is delivered to the Xi register in a normal manner. Subsequent use of this result as an operand in a floating point unit will, however, result in overflow detection. If the exponent computation in this instruction results in exactly +1777 octal and the alternate output path is used, the exponent delivered to the Xi register will be reduced one count and the result will be in floating point range.

Complete overflow

A complete overflow occurs for this instruction whenever the exponent computation results in an exponent greater than +1777 octal. This situation is sensed as a special case, and a complete overflow word with proper sign, overflow exponent, and zero coefficient is delivered to the Xi register. The coefficient calculation is ignored for this case, and the overflow condition flag is set in the PSD register.

Partial underflow

A partial underflow occurs for this instruction whenever the exponent computation results in exactly -1776 octal and the alternate output path is used from the double precision adder. In this case the exponent delivered to the Xi register is reduced one count and creates an underflow exponent with a valid coefficient. There are no condition flags set in the PSD register for this case. Subsequent use of this result in a floating point unit may, however, result in underflow detection.

Complete underflow

A complete underflow occurs for this instruction whenever the exponent computation results in less than -1776 octal. This situation is sensed as a special case, and a complete zero word with proper sign is delivered to the Xi register. The coefficient calculation is ignored in this case, and the underflow condition flag is set in the PSD register.

On operand indefinite

If either operand is indefinite, or if both operands are indefinit , the result is indefinite. The operand coefficients are ignored in this case, and the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

One operand overflow

If one operand has an overflow exponent and the other operand is in floating point range, or if both operands have overflow exponents, the result is a complete overflow word delivered to the Xi register. The coefficients of the operands are ignored in this case, and the result has a zero coefficient. The sign of the result is calculated in the same manner as for operands in range. The overflow condition flag is set in the PSD register for this case.

One operand underflow

If one operand has an underflow exponent and the other operand is in floating point range, or if both operands have underflow exponents, the result is a complete underflow word delivered to the Xi register. The coefficients of the operands are ignored in this case, and the result has a zero coefficient. The sign of the result is calculated in the same manner as for operands in range. The underflow condition flag is set in the PSD register for this case.

Underflow times overflow

If one operand has an underflow exponent and the other operand has an overflow exponent the result is indefinite. The operand coefficients are ignored in this case, and the word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

43ijk Form mask jk

This instruction causes the shift unit to generate a masking word using the j and k designators as parameters. There are no operands read from operating registers for this instruction. The j and k designators are treated as a single six bit quantity to designate the width of the masking field. A field of ones, beginning at the highest order end of the word, is extended downward on a background of zeros. The completed masking word consists of one bits in the highest order jk bit positions of the word and zero bits in the remainder of the word. This masking word is then delivered to the Xi register.

Sample parameters: j = 2
k = 4
(Xi) = 7777 7760 0000 0000 0000

This instruction is intended for generating variable width masks for logical operators. This instruction together with a shift instruction will generally create an arbitrary field mask faster than reading a pre-generated mask from storage.

Issue conditions

Xi register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 43 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Set Xi reservation flag.
Set go shift flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from shift unit to Xd register.
Clear Xd reservation flag.
Clear go shift flag.

Special situations

Designators j and k are zero

If the j and k designators are zero a word containing all zero bits is written into the Xi register. The timing for this case is the same as the timing for the general case.

Quantity jk greater than 60 decimal

If the quantity jk is greater than 60 decimal a word containing all one bits is written into the Xi register. Any jk value of 60 or greater will cause the same result.

44ijk Floating divide

This instruction causes the divide unit to read operands from two X registers, operate upon them to form a floating point quotient, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). These operands are assumed to be numbers in floating point format. The result of dividing (Xj) by (Xk) is delivered to the Xi register. If both operands are normalized the quotient will also be normalized. The remainder from the division process is discarded.

The operands are not rounded in this operation. The two operands are unpacked from floating point format. The exponents are subtracted with a correction factor to determine the exponent for the result. The coefficient from (Xj) is positioned in a dividend register. The coefficient from (Xk) is trial subtracted repeatedly from the dividend and the dividend shifted to form the quotient bits. The quotient bits are assembled in a quotient register. When 48 bits of the quotient have been assembled they are packed with the result exponent into floating point format and delivered to the Xi register.

If the dividend is not normalized the quotient may not be normalized. The quotient will be correct, however, even though there may be leading zeros in the coefficient. If the divisor is not normalized the quotient may be incorrect. If the coefficient for (Xj), considered as an integer, is larger than the coefficient for (Xk) by a factor of 2 or more, then the quotient will be incorrect. This situation is detected and an error flag set when it occurs.

This instruction is intended for use in floating point calculations where rounding of operands is not desired. In multiple precision division this instruction must be followed by a multiplication of quotient by divisor and subtracted from dividend in order to reconstruct the remainder.

Issue conditions

Xi register free.
Xj register free.
Xk register free.
Divide unit free.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register 19 clock periods after the instruction issues. The Xi register will be reserved for the 19 clock periods from issue to delivery of data. The divide unit will be free 17 clock periods after this instruction issues. The command timing for this instruction is listed below.

44 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the divide unit.
Transmit (Xk) to the divide unit.
Set Xi reservation flag.
Set divide busy flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Enter (Xj) coefficient in dividend register.

First trial subtraction.
Sense quotient overflow.

CP13 Transmit block SAS issue signal to SCM control.

CP17 Clear divide busy flag.

CP18 Last trial subtraction.
Enter last quotient bits in quotient register.

CP19 Transmit result from divide unit to Xd register.
Clear Xd reservation flag.

Special situations

Quotient is incorrect

If the divisor is not normalized and the dividend coefficient is larger than the divisor by a factor of two or more, the quotient coefficient will be incorrect. The quotient is disregarded in this case, and the word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

Partial overflow

A partial overflow occurs for this instruction whenever the exponent computation results in exactly +1777 octal. There are no error condition flags set in the PSD register for this case, and the result is delivered to the Xi register in a normal manner. Subsequent use of this result as an operand in a floating point unit will, however, result in overflow detection.

Complete overflow

A complete overflow occurs for this instruction whenever the exponent computation results in an exponent greater than +1777 octal. This situation is sensed as a special case, and a complete overflow word with proper sign, overflow exponent, and zero coefficient is delivered to the Xi register. The coefficient calculation is ignored for this case, and the overflow condition flag is set in the PSD register.

Partial underflow

A partial underflow occurs for this instruction whenever the exponent computation results in exactly -1777 octal. There are no error condition flags set in the PSD register for this case, and the result is delivered to the Xi register in a normal manner. Subsequent use of this result as an operand in a floating point unit may, however, result in underflow detection.

Complete underflow

A complete underflow occurs for this instruction whenever the exponent computation results in an exponent less than -1777 octal. This situation is sensed as a special case, and a complete zero word with proper sign is delivered to the Xi register. The coefficient calculation is ignored in this case, and the underflow condition flag is set in the PSD register.

One operand indefinite

If either operand is indefinite, or if both operands are indefinite, the result is indefinite. The operand coefficients are ignored in this case, and the resulting word delivered to the Xi register is positive indefinite with a zero coefficient. The indefinite condition flag is set in the PSD register for this case.

is overflow w

If (Xj) has an overflow exponent and (Xk) is in floating point range or has an underflow exponent, the result is a complete overflow word delivered to the Xi register. The coefficients of the operands are ignored in this case, and the result has a zero coefficient. The sign of the result is calculated in the same manner as for operands in range. The overflow condition flag is set in the PSD register for this case.

is underflow word

If (Xj) has an underflow exponent and (Xk) is in floating point range or has an overflow exponent, the result is a complete underflow word delivered to the Xi register. The coefficients of the operands are ignored in this case, and the result is a zero word. The sign of the result is calculated in the same manner as for operands in range. The underflow condition flag is set in the PSD register for this case.

is overflow word

If (Xk) has an overflow exponent and (Xj) is in floating point range the result is a complete underflow word delivered to the Xi register. The coefficients of the operands are ignored in this case, and the result is a zero word. The sign of the result is calculated in the same manner as for operands in range. The underflow condition flag is set in the PSD register for this case.

(Xk) is underflow word

If (Xk) has an underflow exponent and (Xj) is in floating point range the result is a complete overflow word delivered to the Xi register. The coefficients of the operands are ignored in this case, and the result has a zero coefficient. The sign of the result is calculated in the same manner as for operands in range. The overflow condition flag is set in the PSD register for this case.

Overflow divided by overflow
Underflow divided by underflow

These combinations of operand exponents result in a positive indefinite word with a zero coefficient delivered to the Xi register. The indefinite condition flag is set in the PSD register for this case.

45ijk Round floating divide

This instruction causes the divide unit to read operands from two X registers, operate upon them to form a rounded floating point quotient, and deliver this result to a third X register. The operands for this instruction are (Xj) and (Xk). These operands are assumed to be numbers in floating point format. The result of dividing (Xj) by (Xk) is delivered to the Xi register. If both operands are normalized the quotient will also be normalized. The remainder from the division process is discarded.

The two operands are unpacked from floating point format in this operation. The exponents are subtracted with a correction factor to determine the exponent for the result. The coefficient from (Xj) is positioned in a dividend register. This quantity is modified by adding a round bit just below the lowest order bit of the coefficient from (Xj). This round bit has the effect of increasing the magnitude of the dividend by one half count. The coefficient from (Xk) is trial subtracted repeatedly from the dividend and the dividend shifted to form the quotient bits. The quotient bits are assembled in a quotient register. When 48 bits of the quotient have been assembled they are packed with the result exponent into floating point format and delivered to the Xi register.

If the dividend is not normalized the quotient may not be normalized. The quotient will be correct, however, even though there may be leading zeros in the coefficient. If the divisor is not normalized the quotient may be incorrect. If the coefficient for (Xj), considered as an integer, is larger than the coefficient for (Xk) by a factor of 2 or more, then the quotient will be incorrect. This situation is detected and an error flag set when it occurs.

This instruction is intended for use in single precision floating point calculations where rounding of operands is desired to reduce truncation errors.

Rounding

The rounding step in this instruction occurs in the dividend register just prior to the first trial subtraction. A round bit is added to the dividend which has the effect of increasing the dividend by one half count. The effect this has on the quotient will vary depending on the value of the divisor and on the truncation point in the quotient. If the dividend is smaller than the divisor the quotient will be truncated one bit position lower than if the dividend is equal to, or larger than, the divisor. These effects cause the rounding to vary in the quotient from a value of 1/4 of the least significant bit in the result to almost one. The average rounding bias over the entire range of coefficient values is zero. The bias in a narrow range of coefficient values varies in two dimensions as shown in the table below.

The table below shows the amount of bias the rounding operation introduces in the quotient for each of 16 fields. Each number in the matrix is an average for the associated field and is expressed as a decimal fraction of the least significant bit position in the result coefficient. The vertical coordinate is the leading octal digit in the divisor. The horizontal coordinate is the leading octal digit in the dividend.

Round bias for normalized operands

	4	5	6	7
4	+.167	-.055	-.055	-.055
5	+.226	+.044	-.137	-.137
5	+.116	+.116	-.038	-.192
7	+.034	+.034	+.034	-.100

Largest positive round error = +.9999
Largest negative round error = -.75
Average round error = 0.00

Issue conditions

Xi register is free.
Xj register is free.
Xk register is free.
Divide unit is free.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register 19 clock periods after the instruction issues. The Xi register will be reserved for the 19 clock periods from issue to delivery of data. The divide unit will be free 17 clock periods after this instruction issues. The command timing for this instruction is the same as the command timing for the 44 instruction.

Special situations

The special situations for this instruction are the same as the special situations listed for the 44 instruction.

46000 Pass

This instruction code causes no action in any functional unit. It is used to fill program instruction words where necessary to match jump destinations with word boundaries. The i, j, and k designators are normally zero in this instruction. These designators are ignored, however, and a nonzero value will have no effect on the instruction issue.

Issue conditions

None.

Execution time

No functional unit is involved in the execution of this instruction. The instruction requires one clock period to issue. Command timing for this instruction is listed below.

CP00 46 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.

Special situations

None.

47i0k Population count

This instruction causes the population count unit to read one operand from register Xk, count the number of one bits in this word, and store this count in the Xi register. The word delivered to the Xi register is in positive integer format. If (Xk) is a word of all ones a count of 60 decimal is delivered to the Xi register. If (Xk) is a word of all zeros a zero word is delivered to the Xi register.

This instruction is intended for use in data processing where a degree of coincidence is desired.

Issue conditions

Xi register is free.
Xk register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. The command timing for this instruction is listed below.

CP00 47 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xk) to the pop count unit.
Set Xi reservation flag.
Set go pop count flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from the pop count unit to the Xd register.
Clear Xd reservation flag.
Clear go pop count flag.

Special situations

Designator j not zero

The j designator in this instruction is normally zero. This designator is ignored, however, in the execution of the instruction, and a nonzero value will have no effect on the results.

Designators i and k have the same value

The i and k designators may have the same value in this instruction. In this case the operand is read from the designated X register and the count then stored back in this same X register. The timing for this case is the same as the timing for the general case.

50ijx	xxxxx

Increment A + K to A

This is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction causes the increment unit to read an operand from the Aj register, form the sum (Aj) + K, and deliver this result to the Ai register. In addition, if the i designator is nonzero, a storage reference is made to SCM using this resulting sum as the relative storage address. The type of storage reference is a function of the i designator value:

i = 0; no storage reference.
i = 1, 2, 3, 4, 5; read from SCM to register Xi.
i = 6, 7; write into SCM from register Xi.

The increment unit forms the sum (Aj) + K in an 18 bit ones complement mode. The resulting sum is simultaneously delivered to the Ai register and to the SAS, if required. The SAS treats this quantity as an 18 bit positive integer address. This address is relative to the beginning of the SCM field for the current program. If the i designator value causes a read from SCM, the result will arrive at the Xi register a minimum of 6 clock periods later than the result delivered to the Ai register. This X register is reserved until the data arrives from storage. If the i designator value causes a write into SCM, the 60 bit word being stored is read from the Xi register into the SWS in the same clock period in which this instruction issues. This X register is therefore not reserved, and may be used for unrelated computation in the next clock period.

This instruction is intended for fetching operands from storage for computation and for delivering results back into storage.

Issue conditions

Xi register is free (i nonzero).
Aj register is free.
Ai register is free.
No SAS backup condition.

Execution time

Designator i = 0

No execution delays are possible after this instruction issues from the CIW register if the i designator is zero. The result will be delivered to the Ai register one clock period after the instruction issues. The Ai register will be reserved for the one clock period from issue to delivery of data. Command timing for this case is listed below.

```
      50 instruction in the upper parcel of the CIW register.
      Instruction issues.
      Transmit the next instruction to upper parcel of CIW register.
      Transmit K to the increment unit.
      Transmit (Aj) to the increment unit.
      Set Ai reservation flag.
      Set go increment flag.

CP01  Next instruction in the upper parcel of the CIW register
      Instruction may issue.
      Transmit data from increment unit to Ad register.
      Clear Ad reservation flag.
      Clear go increment flag.
```

Designator i = 1, 2, 3, 4, 5

These values for the i designator cause a read storage reference in SCM in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. The delivery of data to the Xi register may be delayed in SCM by storage bank conflicts. The Ai register is reserved for the one clock period from issue to delivery of data. The Xi register is reserved for the amount of time from issue to delivery of the word from storage. Command timing for this case is listed below.

```
      50 instruction in the upper parcel of the CIW register.
      Instruction issues.
      Transmit the next instruction to upper parcel of CIW register.
      Transmit K to the increment unit.
      Transmit (Aj) to the increment unit.
      Set Ai reservation flag.
      Set Xi reservation flag.
      Set go increment flag.
```

Next instruction in the upper parcel of the CIW register
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Transmit address + (RAS) to the SAS. Tag for read to X.
Clear go increment flag.

Address leaves SAS for a SCM bank address register.

Begin SCM bank read/write cycle

CP06 SCM data reads to bank operand register.

CP07 Transmit SCM data to Xd register.
Clear Xd reservation flag.

Designator i = 6, 7

These values for the i designator cause a write storage reference in SCM in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. There may be delays in initiating the SCM bank reference to store the data from the Xi register. These delays will not affect the timing for this instruction because the data is read from the Xi register at issue time and is held in the SWS if a delay occurs. The Ai register is reserved for the one clock period from issue to delivery of data. The Xi register is not reserved. The command timing for this case is listed below.

50 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit K to the increment unit.
Transmit (Aj) to the increment unit.
Set Ai reservation flag.
Transmit (Xi) to the SWS.
Set go increment flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Transmit address + (RAS) to the SAS. Tag for write from X.
Clear go increment flag.

Address leaves SAS for a j bank address register.

Begin SCM bank read/write cycle.

CP07 Transmit data from SWS to SCM bank operand register.

Special situations

Last parcel

This instruction normally requires two parcels of an instruction word. If this instruction begins in the first, second, or third parcel of an instruction word the following parcel completes the instruction. If this instruction begins in the last parcel of an instruction word it will not be continued in the following word. In this case the instruction will be executed as if there were a fifth parcel in the instruction word and this parcel contained all zeros.

SCM address out of range

If this instruction makes a storage reference to SCM the address is compared with (FLS) to determine if the reference is within the assigned SCM field. If the address is out of range the SCM direct range condition flag is set in the PSD register. This flag will cause the current program sequence to terminate with an exchange jump to (EEA). If the reference involved reading to an X register, the out of range word addressed will be read into the X register before the interrupt occurs. If the reference involved writing into SCM the memory sequence will be aborted to avoid altering the designated storage quantity.

Designators i and j are the same

The i and j designators may have the same value in this instruction. The initial contents of the designated A register are read to the increment unit and the result stored back into the same A register.

5ijx	xxxxx

Increment B + K to A

This is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction causes the increment unit to read an operand from the Bj register, form the sum (Bj) + K, and deliver this result to the Ai register. In addition, if the i designator is nonzero, a storage reference is made to SCM using this resulting sum as the relative storage address. The type of storage reference is a function of the i designator value:

i = 0; no storage reference.
i = 1, 2, 3, 4, 5; read from SCM to register Xi.
i = 6, 7; write into SCM from register Xi.

The increment unit forms the sum (Bj) + K in an 18 bit ones complement mode. The resulting sum is simultaneously delivered to the Ai register and to the SAS, if required. The SAS treats this quantity as an 18 bit positive integer address. This address is relative to the beginning of the SCM field for the current program. If the i designator value causes a read from SCM, the result will arrive at the Xi register a minimum of 6 clock periods later than the result delivered to the Ai register. This X register is reserved until the data arrives from storage. If the i designator value causes a write into SCM, the 60 bit word being stored is read from the Xi register into the SWS in the same clock period in which this instruction issues. This X register is therefore not reserved, and may be used for unrelated computation in the next clock period.

This instruction is intended for fetching operands from storage for computation and for delivering results back into storage.

Issue conditions

Xi register is free (i nonzero).
Bj register is free.
Ai register is free.
No SAS backup condition.

Execution time

Designator i = 0

No execution delays are possible after this instruction issues from the CIW register if the i designator is zero. The result will be delivered to the Ai register one clock period after the instruction issues. The Ai register will be reserved for the one clock period from issue to delivery of data. Command timing for this case is listed below.

```
51 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit K to the increment unit.
Transmit (Bj) to the increment unit.
Set Ai reservation flag.
Set go increment flag.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Clear go increment flag.
```

Designator i = 1, 2, 3, 4, 5

These values for the i designator cause a read storage reference in SCM in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. The delivery of data to the Xi register may be delayed in SCM by storage bank conflicts. The Ai register is reserved for the one clock period from issue to delivery of data. The Xi register is reserved for the amount of time from issue to delivery of the word from storage. Command timing for this case is listed below.

```
CP00  51 instruction in the upper parcel of the CIW register.
      Instruction issues.
      Transmit the next instruction to upper parcel of CIW register.
      Transmit K to the increment unit.
      Transmit (Bj) to the increment unit.
      Set Ai reservation flag.
      Set Xi reservation flag.
      Set go increment flag.
```

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Transmit address + (RAS) to the SAS. Tag for read to X.
Clear go increment flag.

CP02 Address leaves SAS for a SCM bank address register.

CP03 Begin SCM bank read/write cycle

CP06 SCM data reads to bank operand register.

CP07 Transmit SCM data to Xd register.
Clear Xd reservation flag.

Designator i = 6, 7

These values for the i designator cause a write storage reference in SCM in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. There may be delays in initiating the SCM bank reference to store the data from the Xi register. These delays will not affect the timing for this instruction because the data is read from the Xi register at issue time and is held in the SWS if a delay occurs. The Ai register is reserved for the one clock period from issue to delivery of data. The Xi register is not reserved. The command timing for this case is listed below.

CP00 51 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit K to the increment unit.
Transmit (Bj) to the increment unit.
Set Ai reservation flag.
Transmit (Xi) to the SWS.
Set go increment flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Transmit address + (RAS) to the SAS. Tag for write from X.
Clear go increment flag.

Address leaves SAS for a SCM bank address register

Begin SCM bank read/write cycle.

CP07 Transmit data from SWS to SCM bank operand register.

Special situations

parcel

This instruction normally requires two parcels of an instruction word. If this instruction begins in the first, second, or third parcel of an instruction word the following parcel completes the instruction. If this instruction begins in the last parcel of an instruction word it will not be continued in the following word. In this case the instruction will be executed as if there were a fifth parcel in the instruction word and this parcel contained all zeros.

SCM address out of range

If this instruction makes a storage reference to SCM the address is compared with (FLS) to determine if the reference is within the assigned SCM field. If the address is out of range the SCM direct range condition flag is set in the PSD register. This flag will cause the current program sequence to terminate with an exchange jump to (EEA). If the reference involved reading to an X register, the out of range word addressed will be read into the X register before the interrupt occurs. If the reference involved writing into SCM the memory sequence will be aborted to avoid altering the designated storage quantity.

52ijx	xxxxx

Increment X + K to A

This is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction causes the increment unit to read an operand from the Xj register, form the sum (Xj) + K, and deliver this result to the Ai register. In addition, if the i designator is nonzero, a storage reference is made to SCM using this resulting sum as the relative storage address. The type of storage reference is a function of the i designator value:

i = 0; no storage reference.
i = 1, 2, 3, 4, 5; read from SCM to register Xi
i = 6, 7; write into SCM from register Xi.

The increment unit forms the sum (Xj) + K in an 18 bit ones complement mode. The Xj register is assumed to contain an integer with less than 18 bits of significance. Only the lowest order 18 bits of (Xj) are transmitted to the increment unit. The resulting sum is simultaneously delivered to the Ai register and to the SAS, if required. The SAS treats this quantity as an 18 bit positive integer address. This address is relative to the beginning of the SCM field for the current program. If the i designator value causes a read from SCM, the result will arrive at the Xi register a minimum of 6 clock periods later than the result delivered to the Ai register. This X register is reserved until the data arrives from storage. If the i designator value causes a write into SCM, the 60 bit word being stored is read from the Xi register into the SWS in the same clock period in which this instruction issues. This X register is therefore not reserved, and may be used for unrelated computation in the next clock period.

This instruction is intended for fetching operands from storage for computation and for delivering results back into storage.

Issue conditions

Xi register is free (i nonzero).
Xj register is free.
Ai register is free.
No SAS backup condition.

Execution time

Designator i = 0

No execution delays are possible after this instruction issues from the CIW register if the i designator is zero. The result will be delivered to the Ai register one clock period after the instruction issues. The Ai register will be reserved for the one clock period from issue to delivery of data. Command timing for this case is listed below.

```
      52 instruction in the upper parcel of the CIW register.
      Instruction issues.
      Transmit the next instruction to upper parcel of CIW register.
      Transmit K to the increment unit.
      Transmit (Xj) to the increment unit.
      Set Ai reservation flag.
      Set go increment flag.

CP01  Next instruction in the upper parcel of the CIW register.
      Instruction may issue.
      Transmit data from increment unit to Ad register.
      Clear Ad reservation flag.
      Clear go increment flag.
```

Designator i = 1, 2, 3, 4, 5

These values for the i designator cause a read storage reference in SCM in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. The delivery of data to the Xi register may be delayed in SCM by storage bank conflicts. The Ai register is reserved for the one clock period from issue to delivery of data. The Xi register is reserved for the amount of time from issue to delivery of the word from storage. Command timing for this case is listed below.

```
      52 instruction in the upper parcel of the CIW register.
      Instruction issues.
      Transmit the next instruction to upper parcel of CIW register.
      Transmit K to the increment unit.
      Transmit (Xj) to the increment unit.
      Set Ai reservation flag.
      Set Xi reservation flag.
      Set go increment flag.
```

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Transmit address + (RAS) to the SAS. Tag for read to X.
Clear go increment flag.

CP02 Address leaves SAS for a SCM bank address register

CP03 Begin SCM bank read/write cycle

CP06 SCM data reads to bank operand register

Transmit SCM data to Xd register
Clear Xd reservation flag.

Designator i = 6, 7

These values for the i designator cause a write storage reference in SCM in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. There may be delays in initiating the SCM bank reference to store the data from the Xi register. These delays will not affect the timing for this instruction because the data is read from the Xi register at issue time and is held in the SWS if a delay occurs. The Ai register is reserved for the one clock period from issue to delivery of data. The Xi register is not reserved. The command timing for this case is listed below.

CP00 52 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit K to the increment unit.
Transmit (Xj) to the increment unit.
Set Ai reservation flag.
Transmit (Xi) to the SWS.
Set go increment flag.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Transmit address + (RAS) to the SAS. Tag for write from X.
Clear go increment flag.

Special situations

Last parcel

This instruction normally requires two parcels of an instruction word. If this instruction begins in the first, second, or third parcel of an instruction word the following parcel completes the instruction. If this instruction begins in the last parcel of an instruction word it will not be continued in the following word. In this case the instruction will be executed as if there were a fifth parcel in the instruction word and this parcel contained all zeros.

SCM address out of range

If this instruction makes a storage reference to SCM the address is compared with (FLS) to determine if the reference is within the assigned SCM field. If the address is out of range the SCM direct range condition flag is set in the PSD register. This flag will cause the current program sequence to terminate with an exchange jump to (EEA). If the reference involved reading to an X register, the out of range word addressed will be read into the X register before the interrupt occurs. If the reference involved writing into SCM the memory sequence will be aborted to avoid altering the designated storage quantity.

Designators i and j are the same

The i and j designators may have the same value in this instruction. The initial contents of the designated X register are read to the increment unit. The resulting sum may then be used as an address for a SCM reference which reads a word from storage into this same X register.

(Xj) has more than 18 significant bits

If (Xj) is not an integer with less than 18 bits of significance, the lowest order 18 bits are extracted and treated as an integer. The higher order bits in (Xj) are ignored.

53ijk Increment X + B to A

This instruction causes the increment unit to read operands from the Xj register and the Bk register, form the sum (Xj) + (Bk), and deliver this result to the Ai register. In addition, if the i designator is nonzero, a storage reference is made to SCM using this resulting sum as the relative storage address. The type of storage reference is a function of the i designator value:

i = 0; no storage reference.
i = 1, 2, 3, 4, 5; read from SCM to register Xi.
i = 6, 7; write into SCM from register Xi.

The increment unit forms the sum (Xj) + (Bk) in an 18 bit ones complement mode. The Xj register is assumed to contain an integer with less than 18 bits of significance. Only the lowest order 18 bits of (Xj) are transmitted to the increment unit. The resulting sum is delivered simultaneously to the Ai register and to the SAS, if required. The SAS treats this quantity as an 18 bit positive integer address. This address is relative to the beginning of the SCM field for the current program. If the i designator value causes a read from SCM, the result will arrive at the Xi register a minimum of six clock periods later than the result delivered to the Ai register. This X register is reserved until the data arrives from storage. If the i designator value causes a write into SCM, the 60 bit word being stored is read from the Xi register into the SWS in the same clock period in which this instruction issues. This X register is therefore not reserved, and may be used for unrelated computation in the next clock period.

This instruction is intended for fetching operands from storage for computation and for delivering results back into storage.

Issue conditions

Xi register is free (i nonzero).
Xj register is free.
Bk register is free.
Ai register is free.
No SAS backup condition.

Execution time

Designator i

No execution delays are possible after this instruction issues from the CIW register if the i designator is zero. The result will be delivered to the Ai register one clock period after the instruction issues. The Ai register will be reserved for the one clock period from issue to delivery of data. Command timing for this case is listed below.

```
53 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the increment unit.
Transmit (Bk) to the increment unit.
Set Ai reservation flag.
Set go increment flag.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Clear go increment flag.
```

Designator i = 1, 2, 3, 4, 5

These values for the i designator cause a read storage reference in SCM in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. The delivery of data to the Xi register may be delayed in SCM by storage bank conflicts. The Ai register is reserved for the one clock period from issue to delivery of data. The Xi register is reserved for the amount of time from issue to delivery of the word from storage. Command timing for this case is listed below.

```
CP00  53 instruction in the upper parcel of the CIW register.
      Instruction issues.
      Transmit the next instruction to upper parcel of CIW register.
      Transmit (Xj) to the increment unit.
      Transmit (Bk) to the increment unit.
      Set Ai reservation flag.
      Set Xi reservation flag.
      Set go increment flag.
```

Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Transmit address + (RAS) to the SAS. Tag for read to X.
Clear go increment flag.

Address leaves SAS for a SCM bank address register.

CP03 Begin SCM bank read/write cycle.

CP06 SCM data reads to bank operand register.

Transmit SCM data to Xd register
Clear Xd reservation flag.

Designator i = 6, 7

These values for the i designator cause a write storage reference in SCM in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. There may be delays in initiating the SCM bank reference to store the data from the Xi register. These delays will not affect the timing for this instruction because the data is read from the Xi register at issue time and is held in the SWS if a delay occurs. The Ai register is reserved for the one clock period from issue to delivery of data. The Xi register is not reserved. The command timing for this case is listed below.

CP00 53 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the increment unit.
Transmit (Bk) to the increment unit.
Set Ai reservation flag.
Transmit (Xi) to the SWS.
Set go increment flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Transmit address + (RAS) to the SAS. Tag for write from X.
Clear go increment flag.

CP02 Address [illegible] for SCM bank [illegible] register

CP03 Begin S[illegible] read/w[illegible] cyc[illegible]

CP07 Transmit data from SWS to SCM bank operand register.

Special situations

SCM address out of range

If this instruction makes a storage reference to SCM the address is compared with (FLS) to determine if the reference is within the assigned SCM field. If the address is out of range the SCM direct range condition flag is set in the PSD register. This flag will cause the current program sequence to terminate with an exchange jump to (EEA). If the reference involved reading to an X register, the out of range word addressed will be read into the X register before the interrupt occurs. If the reference involved writing into SCM the memory sequence will be aborted to avoid altering the designated storage quantity.

Designators i and j are the same

The i and j designators may have the same value in this instruction. The initial contents of the designated X register are read to the increment unit. The resulting sum may then be used as an address for a SCM reference which reads a word from storage into this same X register.

(Xj) has more than 18 significant bits

If (Xj) is not an integer with less than 18 bits of significance, the lowest order 18 bits are extracted and treated as an integer. The higher order bits in (Xj) are ignored.

54ijk Increment A + B to A

This instruction causes the increment unit to read operands from the Aj register and the Bk register, form the sum (Aj) + (Bk), and deliver this result to the Ai register. In addition, if the i designator is nonzero, a storage reference is made to SCM using this resulting sum as the relative storage address. The type of storage reference is a function of the i designator value:

i = 0; no storage reference.
i = 1, 2, 3, 4, 5; read from SCM to register Xi.
i = 6, 7; write into SCM from register Xi.

The increment unit forms the sum (Aj) + (Bk) in an 18 bit ones complement mode. The resulting sum is delivered simultaneously to the Ai register and to the SAS, if required. The SAS treats this quantity as an 18 bit positive integer address. This address is relative to the beginning of the SCM field for the current program. If the i designator value causes a read from SCM, the result will arrive at the Xi register a minimum of six clock periods later than the result delivered to the Ai register. This X register is reserved until the data arrives from storage. If the i designator value causes a write into SCM, the 60 bit word being stored is read from the Xi register into the SWS in the same clock period in which this instruction issues. This X register is therefore not reserved, and may be used for unrelated computation in the next clock period.

This instruction is intended for fetching operands from storage for computation and for delivering results back into storage.

Issue conditions

Xi register is free (i nonzero).
Aj register is free.
Bk register is free.
Ai register is free.
No SAS backup condition.

Execution time

Designator i = 0

No execution delays are possible after this instruction issues from the CIW register if the i designator is zero. The result will be delivered to the Ai register one clock period after the instruction issues. The Ai register will be reserved for the one clock period from issue to delivery of data. Command timing for this case is listed below.

CP00 54 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Aj) to the increment unit.
Transmit (Bk) to the increment unit.
Set Ai reservation flag.
Set go increment flag.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Clear go increment flag.

Designator i = 1, 2, 3, 4, 5

These values for the i designator cause a read storage reference in SCM in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. The delivery of data to the Xi register may be delayed in SCM by storage bank conflicts. The Ai register is reserved for the one clock period from issue to delivery of data. The Xi register is reserved for the amount of time from issue to delivery of the word from storage. Command timing for this case is listed below.

54 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Aj) to the increment unit.
Transmit (Bk) to the increment unit.
Set Ai reservation flag.
Set Xi reservation flag.
Set go increment flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Transmit address + (RAS) to the SAS. Tag for read to X.
Clear go increment flag.

CP02 Address leaves SAS for a SCM bank address register.

CP03 Begin SCM bank read/write cycle.

CP06 SCM data reads to bank operand register.

CP07 Transmit SCM data to Xd register.
Clear Xd reservation flag.

Designator i = 6, 7

These values for the i designator cause a write storage reference in SCM in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. There may be delays in initiating the SCM bank reference to store the data from the Xi register. These delays will not affect the timing for this instruction because the data is read from the Xi register at issue time and is held in the SWS if a delay occurs. The Ai register is reserved for the one clock period from issue to delivery of data. The Xi register is not reserved. The command timing for this case is listed below.

CP00 54 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Aj) to the increment unit.
Transmit (Bk) to the increment unit.
Set Ai reservation flag.
Transmit (Xi) to the SWS.
Set go increment flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Transmit address + (RAS) to the SAS. Tag for write from X.
Clear go increment flag.

Address leaves SAS for a SCM bank address register.

Begin SCM bank read/write cycle.

Transmit data from SWS to SCM bank operand register.

Special situations

SCM address out of range

If this instruction makes a storage reference to SCM the address is compared with (FLS) to determine if the reference is within the assigned SCM field. If the address is out of range the SCM direct range condition flag is set in the PSD register. This flag will cause the current program sequence to terminate with an exchange jump to (EEA). If the reference involved reading to an X register, the out of range word addressed will be read into the X register before the interrupt occurs. If the reference involved writing into SCM the memory sequence will be aborted to avoid altering the designated storage quantity.

Designators i and j are the same

The i and j designators may have the same value in this instruction. The initial contents of the designated A register are read to the increment unit and the result stored back into the same A register.

55ijk Increment A - B to A

This instruction causes the increment unit to read operands from the Aj register and the Bk register, form the difference (Aj) - (Bk), and deliver this result to the Ai register. In addition, if the i designator is nonzero, a storage reference is made to SCM using this result as the relative storage address. The type of storage reference is a function of the i designator value:

i = 0; no storage reference.
i = 1, 2, 3, 4, 5; read from SCM to register Xi.
i = 6, 7; write into SCM from register Xi.

The increment unit forms the difference (Aj) - (Bk) in an 18 bit ones complement mode. The result is delivered simultaneously to the Ai register and to the SAS, if required. The SAS treats this quantity as an 18 bit positive integer address. This address is relative to the beginning of the SCM field for the current program. If the i designator value causes a read from SCM, the result will arrive at the Xi register a minimum of six clock periods later than the result delivered to the Ai register. This X register is reserved until the data arrives from storage. If the i designator value causes a write into SCM, the 60 bit word being stored is read from the Xi register into the SWS in the same clock period in which this instruction issues. This X register is therefore not reserved, and may be used for unrelated computation in the next clock period.

This instruction is intended for fetching operands from storage for computation and for delivering results back into storage.

Issue conditions
Execution time
Special situations

The issue conditions, execution time, and special situations for this instruction are the same as those listed for the 54 instruction.

56ijk Increment B + B to A

This instruction causes the increment unit to read operands from the Bj register and the Bk register, form the sum (Bj) + (Bk), and deliver this result to the Ai register. In addition, if the i designator is nonzero, a storage reference is made to SCM using this result as the relative storage address. The type of storage reference is a function of the i designator value:

i = 0; no storage reference.
i = 1, 2, 3, 4, 5; read from SCM to register Xi
i = 6, 7; write into SCM from register Xi.

The increment unit forms the sum (Bj) + (Bk) in an 18 bit ones complement mode. The result is delivered simultaneously to the Ai register and to the SAS, if required. The SAS treats this quantity as an 18 bit positive integer address. This address is relative to the beginning of the SCM field for the current program. If the i designator value causes a read from SCM, the result will arrive at the Xi register a minimum of six clock periods later than the result delivered to the Ai register. This X register is reserved until the data arrives from storage. If the i designator value causes a write into SCM, the 60 bit word being stored is read from the Xi register into the SWS in the same clock period in which this instruction issues. This X register is therefore not reserved, and may be used for unrelated computation in the next clock period.

This instruction is intended for fetching operands from storage for computation and for delivering results back into storage.

Issue conditions

Xi register is free (i nonzero).
Bj register is free.
Bk register is free.
Ai register is free.
No SAS backup condition.

Execution time

Designator i = 0

No execution delays are possible after this instruction issues from the CIW register if the i designator is zero. The result will be delivered to the Ai register one clock period after the instruction issues. The Ai register will be reserved for the one clock period from issue to delivery of data. Command timing for this case is listed below.

CP00 56 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register
Transmit (Bj) to the increment unit.
Transmit (Bk) to the increment unit.
Set Ai reservation flag.
Set go increment flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Clear go increment flag.

Designator i = 1, 2, 3, 4, 5

These values for the i designator cause a read storage reference in SCM in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. The delivery of data to the Xi register may be delayed in SCM by storage bank conflicts. The Ai register is reserved for the one clock period from issue to delivery of data. The Xi register is reserved for the amount of time from issue to delivery of the word from storage. Command timing for this case is listed below.

CP00 56 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Bj) to the increment unit.
Transmit (Bk) to the increment unit.
Set Ai reservation flag.
Set Xi reservation flag.
Set go increment flag.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Transmit address + (RAS) to the SAS. Tag for read to X.
Clear go increment flag.

CP02 Address leaves SAS for a SCM bank address register.

CP03 Begin SCM bank read/write cycle.

SCM data reads to bank operand register.

Transmit SCM data to Xd register.
Clear Xd reservation flag.

Designator i = 6, 7

These values for the i designator cause a write storage reference in SCM in addition to the result delivered to the Ai register. No delays are possible from issue to delivery of data to the Ai register and to the SAS. There may be delays in initiating the SCM bank reference to store the data from the Xi register. These delays will not affect the timing for this instruction because the data is read from the Xi register at issue time and is held in the SWS if a delay occurs. The Ai register is reserved for the one clock period from issue to delivery of data. The Xi register is not reserved. The command timing for this case is listed below.

CP00 56 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Bj) to the increment unit.
Transmit (Bk) to the increment unit.
Set Ai reservation flag.
Transmit (Xi) to the SWS.
Set go increment flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Ad register.
Clear Ad reservation flag.
Transmit address + (RAS) to the SAS. Tag for write from X.
Clear go increment flag.

CP02 Address leaves SAS for a SCM bank address register.

CP03 Begin SCM bank read/write cycle.

CP07 Transmit data from SWS to SCM bank operand register.

Special situations

SCM address out of range

If this instruction makes a storage reference to SCM the address is compared with (FLS) to determine if the reference is within the assigned SCM field. If the address is out of range the SCM direct range condition flag is set in the PSD register. This flag will cause the current program sequence to terminate with an exchange jump to (EEA). If the reference involved reading to an X register, the out of range word addressed will be read into the X register before the interrupt occurs. If the reference involved writing into SCM the memory sequence will be aborted to avoid altering the designated storage quantity.

57ijk Increment B - B to A

This instruction causes the increment unit to read operands from the Bj register and the Bk register, form the difference (Bj) - (Bk), and deliver this result to the Ai register. In addition, if the i designator is nonzero, a storage reference is made to SCM using this result as the relative storage address. The type of storage reference is a function of the i designator value:

i = 0; no storage reference.
i = 1, 2, 3, 4, 5; read from SCM to register Xi.
i = 6, 7; write into SCM from register Xi.

The increment unit forms the difference (Bj) - (Bk) in an 18 bit ones complement mode. The result is delivered simultaneously to the Ai register and to the SAS, if required. The SAS treats this quantity as an 18 bit positive integer address. This address is relative to the beginning of the SCM field for the current program. If the i designator value causes a read from SCM, the result will arrive at the Xi register a minimum of six clock periods later than the result delivered to the Ai register. This X register is reserved until the data arrives from storage. If the i designator value causes a write into SCM, the 60 bit word being stored is read from the Xi register into the SWS in the same clock period in which this instruction issues. This X register is therefore not reserved, and may be used for unrelated computation in the next clock period.

This instruction is intended for fetching operands from storage for computation and for delivering results back into storage.

Issue conditions
Execution time
Special situations

The issue conditions, execution time, and special situations for this instruction are the same as those listed for the 56 instruction.

60ijx	xxxxx

Increment A + K to B

This is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction causes the increment unit to read an operand from the Aj register, form the sum (Aj) + K, and deliver this result to the Bi register.

The increment unit forms the sum (Aj) + K in an 18 bit ones complement mode. This instruction is intended as a vehicle for address modification in the increment registers.

Issue conditions

Bi register is free.
Aj register is free.
B register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Bi register one clock period after the instruction issues. The Bi register will be reserved for the one clock period from issue to delivery of data. Command timing for this instruction is listed below.

CP00 60 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit K to the increment unit.
Transmit (Aj) to the increment unit.
Set Bi reservation flag.
Set go increment flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Be register.
Clear Be reservation flag.
Clear go increment flag.

Special situations

Designator i is zero

If the i designator is zero this instruction becomes a pass instruction. The B0 register is always free and the result stored in the B0 register is discarded.

Last parcel

This instruction normally requires two parcels of an instruction word. If this instruction begins in the first, second, or third parcel of an instruction word the following parcel completes the instruction. If this instruction begins in the last parcel of an instruction word it will not be continued in the following word. In this case the instruction will be executed as if there were a fifth parcel in the instruction word and this parcel contained all zeros.

61ijx	xxxxx

Increment B + K to B

This is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction causes the increment unit to read an operand from the Bj register, form the sum (Bj) + K, and deliver this result to the Bi register. The increment unit forms the sum (Bj) + K in an 18 bit ones complement mode.

Issue conditions

Bi register is free.
Bj register is free.
B register input path will be free in next clock period.
No SAS backup condition.

Execution time

The execution timing for this instruction is the same as that listed for the 60 instruction. The command timing is the same except that the Bj register is used for the operand rather than the Aj register.

Special situations

The special situations for this instruction are the same as those listed for the 60 instruction.

62ijx	xxxxx

Increment X + K to B

This is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction causes the increment unit to read an operand from the Xj register, form the sum (Xj) + K, and deliver this result to the Bi register.

The increment unit forms the sum (Xj) + K in an 18 bit ones complement mode. The Xj register is assumed to contain an integer with less than 18 bits of significance. Only the lowest order 18 bits of (Xj) are transmitted to the increment unit.

This instruction provides the vehicle for entering an increment register with computation results or with an operand read from storage.

Issue conditions

Xj register is free.
Bi register is free.
B register input path will be free in next clock period.
No SAS backup condition.

Execution time

The execution timing for this instruction is the same as that listed for the 60 instruction. The command timing is the same except that the Xj register is used for the operand rather than the Aj register.

Special situations

The special situations for this instruction are the same as those listed for the 60 instruction plus the additional situation listed below.

(Xj) has more than 18 significant bits

If (Xj) is not an integer with less than 18 bits of significance the lowest order 18 bits are extracted and treated as an integer The higher order bits in (Xj) are ignored.

63ijk Increment X + B to B

This instruction causes the increment unit to read operands from the Xj register and the Bk register, form the sum (Xj) + (Bk), and deliver this result to the Bi register. The increment unit forms the sum (Xj) + (Bk) in an 18 bit ones complement mode. The Xj register is assumed to contain an integer with less than 18 bits of significance. Only the lowest order 18 bits of (Xj) are transmitted to the increment unit.

Issue conditions

Xj register is free.
Bk register is free.
Bi register is free.
B register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Bi register one clock period after the instruction issues. The Bi register will be reserved for the one clock period from issue to delivery of data. Command timing for this instruction is listed below.

63 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Xj) to the increment unit.
Transmit (Bk) to the increment unit.
Set Bi reservation flag.
Set go increment flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Be register.
Clear Be reservation flag.
Clear go increment flag.

Special situations

Designator i is zero

If the i designator is zero this instruction becomes a pass instruction. The B0 register is always free, and the result stored in the B0 register is discarded.

(Xj) has more than 18 significant bits

If (Xj) is not an integer with less than 18 bits of significance, the lowest order 18 bits of (Xj) are extracted and treated as an integer. The higher order bits in (Xj) are ignored.

64ijk Increment A + B to B

This instruction causes the increment unit to read operands from the Aj register and the Bk register, form the sum (Aj) + (Bk), and deliver this result to the Bi register. The increment unit forms the sum (Aj) + (Bk) in an 18 bit ones complement mode.

Issue conditions

Aj register is free.
Bk register is free.
Bi register is free.
B register input path will be free in next clock period.
No SAS backup condition.

Execution time

The execution timing for this instruction is the same as that listed for the 63 instruction. The command timing is the same except that the Aj register is used for the operand rather than the Xj register.

Special situations

Designator i is zero

If the i designator is zero this instruction becomes a pass instruction. The B0 register is always free, and the result stored in the B0 register is discarded.

65ijk Increment A - B to B

This instruction causes the increment unit to read operands from the Aj register and the Bk register, form the difference (Aj) - (Bk), and deliver this result to the Bi register. The increment unit forms the difference (Aj) - (Bk) in an 18 bit ones complement mode.

Issue conditions

Aj register is free.
Bk register is free.
Bi register is free.
B register input path will be free in next clock period.
No SAS backup condition.

Execution time

The execution timing for this instruction is the same as that listed for the 63 instruction. The command timing is the same except that the Aj register is used for the operand rather than the Xj register.

Special situations

Designator i is zero

If the i designator is zero this instruction becomes a pass instruction. The B0 register is always free, and the result stored in the B0 register is discarded.

66ijk Increment B + B to B

This instruction causes the increment unit to read operands from the Bj register and the Bk register, form the sum (Bj) + (Bk), and deliver this result to the Bi register. The increment unit forms the sum (Bj) + (Bk) in an 18 bit ones complement mode.

Issue conditions

Bj register is free.
Bk register is free.
Bi register is free.
B register input path will be free in next clock period.
No SAS backup condition.

Execution time

The execution timing for this instruction is the same as that listed for the 63 instruction. The command timing is the same except that the Bj register is used for the operand rather than the Xj register.

Special situations

Designator i is zero

If the i designator is zero this instruction becomes a pass instruction. The B0 register is always free, and the result stored in the B0 register is discarded.

67ijk Increment B - B to B

This instruction causes the increment unit to read operands from the Bj register and the Bk register, form the difference (Bj) - (Bk), and deliver this result to the Bi register. The increment unit forms the difference (Bj) - (Bk) in an 18 bit ones complement mode.

Issue conditions

Bj register is free.
Bk register is free.
Bi register is free.
B register input path will be free in next clock period.
No SAS backup condition.

Execution time

The execution timing for this instruction is the same as that listed for the 63 instruction. The command timing is the same except that the Bj register is used for the operand rather than the Xj register.

Special situations

Designator i is zero

If the i designator is zero this instruction becomes a pass instruction. The B0 register is always free, and the result stored in the B0 register is discarded.

70ijx	xxxxx

Increment A + K to X

This is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction causes the increment unit to read an operand from the Aj register, form the sum (Aj) + K, and deliver this result to the Xi register. The increment unit forms the sum (Aj) + K in an 18 bit ones complement mode. The resulting 18 bit quantity is sign extended for the 60 bit Xi register by copying the highest order bit of the result into the upper 42 bit positions in the Xi register.

Issue conditions

Aj register is free.
Xi register is free.
X register input path will be free in next clock period
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. Command timing for this instruction is listed below.

CP00 70 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit K to the increment unit.
Transmit (Aj) to the increment unit.
Set Xi reservation flag.
Set go increment flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Xd register.
Clear Xd reservation flag.
Clear go increment flag.

Special situations

Special situations for this instruction are the same as those listed for the 71 instruction.

71ijx	xxxxx

Increment B + K to X

This is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction causes the increment unit to read an operand from the Bj register, form the sum (Bj) + K, and deliver this result to the Xi register. The increment unit forms the sum (Bj) + K in an 18 bit ones complement mode. The resulting 18 bit quantity is sign extended for the 60 bit Xi register by copying the highest order bit of the result into the upper 42 bit positions in the Xi register.

Issue conditions

Bj register is free.
Xi register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

The execution timing for this instruction is the same as that listed for the 70 instruction. The command timing is the same except that the Bj register is used for the operand rather than the Aj register.

Special situations

Last parcel

This instruction normally requires two parcels of an instruction word. If this instruction begins in the first, second, or third parcel of an instruction word the following parcel completes the instruction. If this instruction begins in the last parcel of an instruction word it will not be continued in the following word. In this case the instruction will be executed as if there were a fifth parcel in the instruction word and this parcel contained all zeros.

72ijx	xxxxx

Increment X + K to X

This is a two parcel instruction in which the lower order 18 bits are used as an operand K. This instruction causes the increment unit to read an operand from the Xj register, form the sum (Xj) + K, and deliver this result to the Xi register. The increment unit forms the sum (Xj) + K in an 18 bit ones complement mode. The Xj register is assumed to contain an integer with less than 18 bits of significance. Only the lowest order 18 bits of (Xj) are transmitted to the increment unit. The 18 bit result from the increment unit is sign extended for the 60 bit Xi register by copying the highest order bit of the result into the upper 42 bit positions in the Xi register.

Issue conditions

Xj register is free.
Xi register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

The execution timing for this instruction is the same as that listed for the 70 instruction. The command timing is the same except that the Xj register is used for the operand rather than the Aj register.

Special situations

The special situations for this instruction are the same as that listed for the 71 instruction plus the additional situation listed below.

(Xj) has more than 18 significant bits

If (Xj) is not an integer with less than 18 bits of significance, the lowest order 18 bits are extracted and treated as an integer. The higher order bits in (Xj) are ignored.

73ijk Increment X + B to X

This instruction causes the increment unit to read operands from the Xj register and the Bk register, form the sum (Xj) + (Bk), and deliver this result to the Xi register. The increment unit forms the sum (Xj) + (Bk) in an 18 bit ones complement mode. The Xj register is assumed to contain an integer with less than 18 bits of significance. Only the lowest order 18 bits of (Xj) are transmitted to the increment unit. The 18 bit result from the increment unit is sign extended for the 60 bit Xi register by copying the highest order bit of the result into the upper 42 bit positions in the Xi register.

Issue conditions

Xj register is free.
Bk register is free.
Xi register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

The execution timing for this instruction is the same as that listed for the 74 instruction. The command timing is the same except that the Xj register is used for the operand rather than the Aj register.

Special situations

(Xj) has more than 18 significant bits

If (Xj) is not an integer with less than 18 significant bits, the lowest order 18 bits are extracted and treated as an integer. The higher order bits in (Xj) are ignored.

74ijk Increment A + B to X

This instruction causes the increment unit to read operands from the Aj register and the Bk register, form the sum (Aj) + (Bk), and deliver this result to the Xi register. The increment unit forms the sum (Aj) + (Bk) in an 18 bit ones complement mode. The 18 bit result from the increment unit is sign extended for the 60 bit Xi register by copying the highest order bit of the result into the upper 42 bit positions in the Xi register.

Issue conditions

Aj register is free.
Bk register is free.
Xi register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. Command timing for this instruction is listed below.

CP00 74 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Aj) to the increment unit.
Transmit (Bk) to the increment unit.
Set Xi reservation flag.
Set go increment flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Xd register.
Clear Xd reservation flag.
Clear go increment flag.

Special situations

None.

75ijk Increment A - B to X

This instruction causes the increment unit to read operands from the Aj register and the Bk register, form the difference (Aj) - (Bk), and deliver this result to the Xi register. The increment unit forms the difference (Aj) - (Bk) in an 18 bit ones complement mode. The 18 bit result from the increment unit is sign extended for the 60 bit Xi register by copying the highest order bit of the result into the upper 42 bit positions in the Xi register.

Issue conditions

Aj register is free.
Bk register is free.
Xi register is free.
X register input path will be free in next clock period
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. Command timing for this instruction is listed below.

CP00 75 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Aj) to the increment unit.
Transmit (Bk) to the increment unit.
Set Xi reservation flag.
Set go increment flag.

CP01 Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Xd register.
Clear Xd reservation flag.
Clear go increment flag.

Special situations

None.

76ijk Increment B + B to X

This instruction causes the increment unit to read operands from the Bj register and the Bk register, form the sum (Bj) + (Bk), and deliver this result to the Xi register. The increment unit forms the sum (Bj) + (Bk) in an 18 bit ones complement mode. The 18 bit result from the increment unit is sign extended for the 60 bit Xi register by copying the highest order bit of the result into the upper 42 bit positions in the Xi register.

Issue conditions

Bj register is free.
Bk register is free.
Xi register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. Command timing for this instruction is listed below.

CP00 76 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Bj) to the increment unit
Transmit (Bk) to the increment unit.
Set Xi reservation flag.
Set go increment flag.

Next instruction in the upper parcel of the CIW register.
Instruction may issue.
Transmit data from increment unit to Xd register.
Clear Xd reservation flag.
Clear go increment flag.

Special situations

None.

77ijk Increment B - B to X

This instruction causes the increment unit to read operands from the Bj register and the Bk register, form the difference (Bj) - (Bk), and deliver this result to the Xi register. The increment unit forms the difference (Bj) - (Bk) in an 18 bit ones complement mode. The 18 bit result from the increment unit is sign extended for the 60 bit Xi register by copying the highest order bit of the result into the upper 42 bit positions in the Xi register.

Issue conditions

Bj register is free.
Bk register is free.
Xi register is free.
X register input path will be free in next clock period.
No SAS backup condition.

Execution time

No execution delays are possible after this instruction issues from the CIW register. The result will be delivered to the Xi register one clock period after the instruction issues. The Xi register will be reserved for the one clock period from issue to delivery of data. Command timing for this instruction is listed below.

77 instruction in the upper parcel of the CIW register.
Instruction issues.
Transmit the next instruction to upper parcel of CIW register.
Transmit (Bj) to the increment unit.
Transmit (Bk) to the increment unit.
Set Xi reservation flag.
Set go increment flag.

Next instruction in the upper parcel of the CIW register
Instruction may issue.
Transmit data from increment unit to Xd register.
Clear Xd reservation flag.
Clear go increment flag.

Special situations

None.

4 PPU DESCRIPTION

PART 4: PPU DESCRIPTION

Introduction

This part of the reference manual describes the structure of a PPU in detail and describes the characteristics of the full duplex communication channels which interconnect the PPUs. The PPUs are constructed of the same type of circuits described in part two of this manual for the CPU. The individual units are composed of registers which are interconnected by static networks to perform the desired logical functions. Those PPUs which reside in the main frame cabinet use the same clock timing mechanism as is used in the CPU. The clock period is 27.5 nanoseconds and the clock pulse duration is eight nanoseconds as illustrated on page 2-0. Those PPUs which reside outside the main frame cabinet have their own clock source. This clock source has the same period and waveform as the main frame clock but is not synchronous with the main frame clock. For this reason the PPUs have synchronizing networks for all data transmitted over a communication channel.

The structure of a PPU is illustrated in figure 4-1 on the following page. There are eight input data paths and eight output data paths connecting the PPU to other devices. The PPU has an internal storage capacity of 4096 twelve bit words. This may be arbitrarily divided between program storage and data storage. The storage is arranged in two independent banks of core memory each with a cycle time of 275 nanoseconds. Operation of the PPU is controlled by a stored program which is sequentially executed in a one address mode. All manipulative operations are performed in an 18 bit accumulator register. Arithmetic is binary in a ones complement mode. The program instructions make use of 64 index registers which are located in the lowest order 64 words of the core memory. Address arithmetic involving these registers is performed in a separate address arithmetic unit which adds two addresses in a 12 bit ones complement mode.

The PPU instructions and input/output channels are designed to permit the PPU to control directly peripheral equipment devices with a minimum of intervening circuits. A modest amount of character conversion and formatting of data may be performed in the PPU before data is transmitted to the CPU. In addition, the PPU may be programed to perform the synchronizing function required in interfacing an electromechanical device to the CPU. In this mode the PPU is generally dedicated to one or a small number of specific devices such as printers, card readers, tape units, disk files, etc.

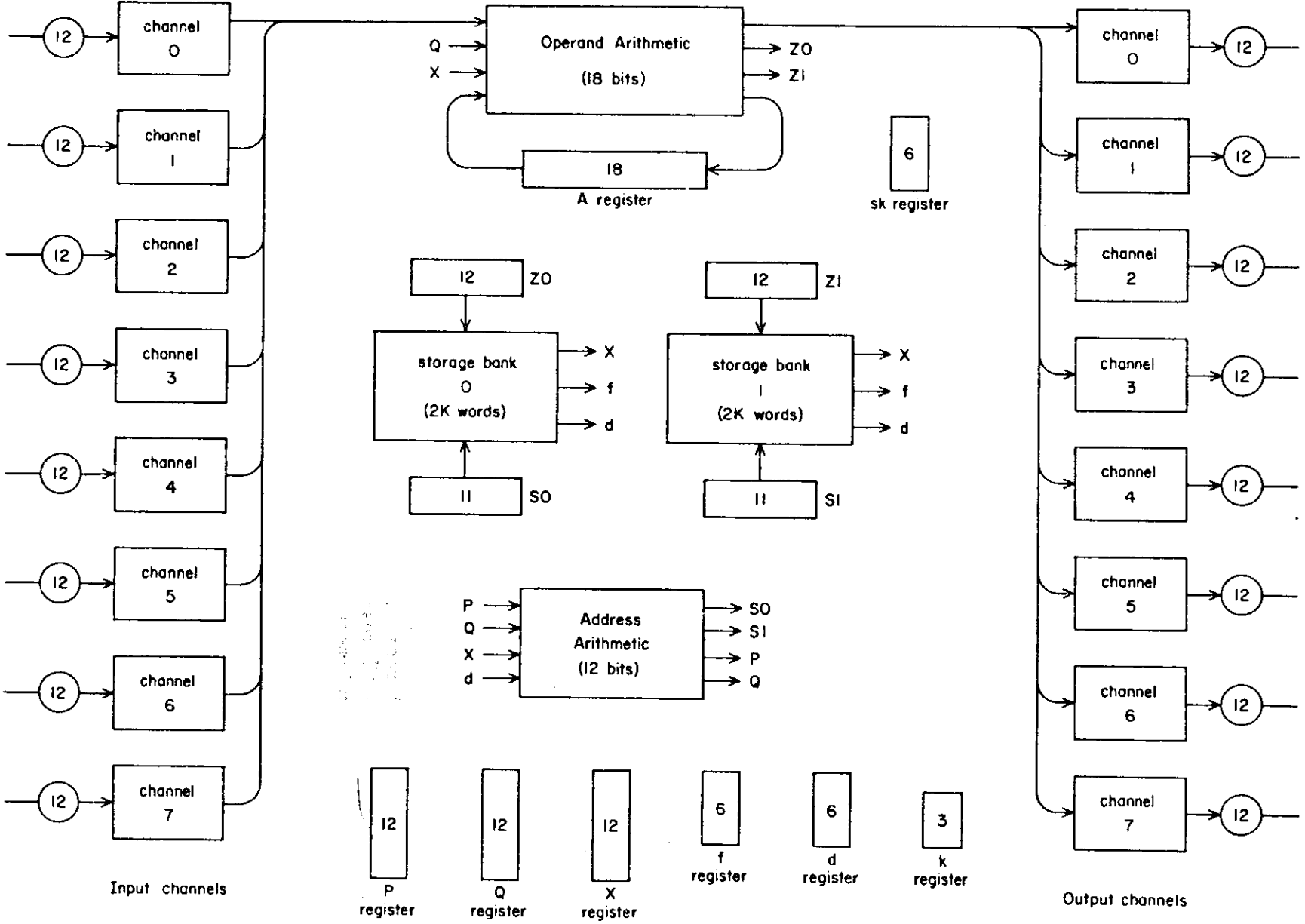


Fig. 4–1 PPU Organization

Operand Arithmetic

Operand arithmetic in a PPU involves three registers and a series of static networks. The principal operand register is the A register. This is an 18 bit register which holds an operand from the execution of one PPU instruction to the next. It is the only register, other than the program address register, which provides continuity in program execution. One operand in an arithmetic function is always contained in the A register. The other two registers involved in arithmetic operations are the X register and the Q register. These registers each contain 12 bits of data. Only the lowest order six bits of the Q register content are involved in arithmetic operations. The X register is the storage operand register and generally contains the lowest order 12 bits of an arithmetic operand. The Q register is involved only in those instructions with an 18 bit storage operand. In those cases the lowest order six bits of the Q register are loaded with the highest order six bits of the 18 bit operand.

The operand arithmetic networks are illustrated in figure 4-2 on the following page. The A register is always the destination of the arithmetic result. The various programable functions are divided into several static networks. Each network has several modes of operation. The various combinations are listed later in this section for the numerous arithmetic modes. These modes fall into three general categories. One category involves six bit storage operands, where the operand generally is contained in the program instruction word. In these cases the lowest order six bits of the A register are operated upon by the storage operand. The highest order 12 bits of the A register are operated upon by 12 zero bits or 12 one bits as specified by the program instruction. A second category involves 12 bit storage operands. In these cases the highest order six bits of the A register are operated upon by six zero bits or six one bits as specified by the program instruction. In the last category the entire 18 bit A register content is operated upon by an 18 bit storage operand, the highest order six bits residing in the Q register, and the lowest order 12 bits residing in the X register.

A separate shift network is illustrated in figure 4-2. This network provides a one place shift of the A register content, either to the left or to the right. The A register content is shifted a number of positions by repeatedly shifting one bit position each clock period. This is the only arithmetic operation in which the execution time is a variable function of the operand data.

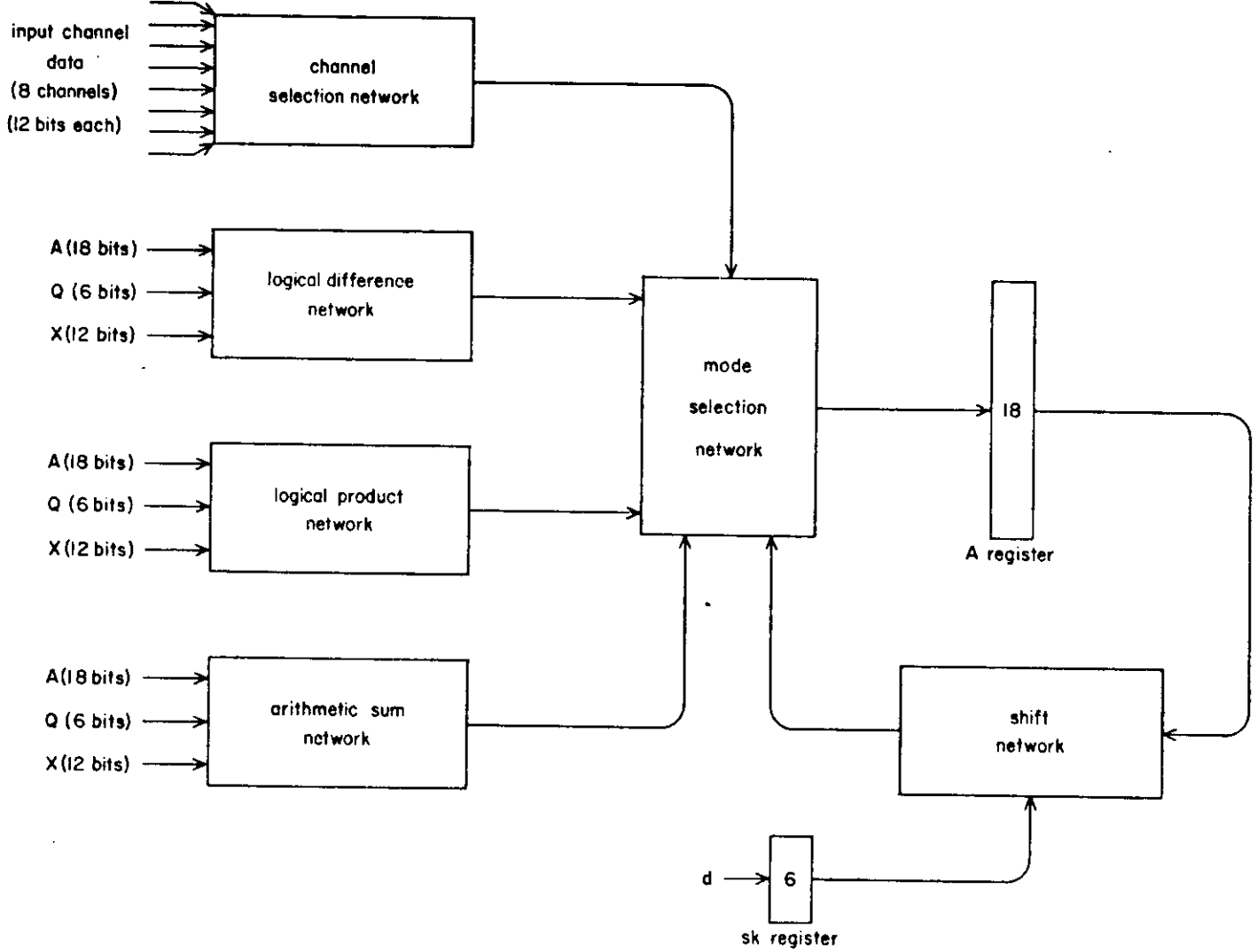


Fig. 4-2 Operand Arithmetic

A register

The A register is an 18 bit clear/enter type register with gated clock pulse control. This register is cleared, and new data is entered, only at the end of those clock periods in which the control condition defined below exists. The A register is generally cleared and entered with new data once in the execution of a program instruction. On some instructions the A register content is altered twice. In block input instructions, or block output instructions, the A register is used to count the length of the block. In these cases the A register is cleared and entered with a new count as each word in the block is transmitted. In the shift A instruction the A register is cleared and entered with a shifted copy of its old value the number of times specified in the shift count.

The quantity entered in the A register is specified in the list of different control conditions for operand arithmetic. The condition for clearing the A register and entering a new value is as follows:

Condition: lowest order five bits in (sk) $\neq$ 0
or: (f) = 10,11,12,13,14,15,16,17 & GRF
or: (f) = 20,21,22,23 & (k) = 1 & GRF
or: (f) = 30,31,32,33,35,36,37 & (k) = 1 & GRF
or: (f) = 36,37,46,47,56,57 & (k) = 0 & GRF
or: (f) = 40,41,42,43,45,46,47 & (k) = 2 & GRF
or: (f) = 50,51,52,53,55,56,57 & (k) = 3 & GRF
or: (f) = 70 & GRF
or: (f) = 71,73 & (k) = 1 & no "A zero" & GRF
or: (f) = 71 & (k) = 2 & no "A zero" & no IRF & GRF
or: (f) = 73 & (k) = 2 & no "A zero" & GRF

"Plus one" condition

This control condition is generated in the operand arithmetic unit to clear the A register and enter a value of plus one. This condition is defined as follows:

Condition: (f) = 36,46,56 & (k) = 0

"Minus one" condition

This control condition is generated in the operand arithmetic unit to clear the A register and enter a value of minus one. This is the same as entering an octal value of 777776. This condition is defined as follows:

Condition: (f) = 37,47,57 & (k) = 0

"A - 1" condition

This control condition is generated in the operand arithmetic unit during the execution of block input, or block output, instructions. The 18 bits of data entered in the A register are formed by the ones complement sum of (A) + 777776. This condition is defined as follows:

Condition: (f) = 71,73

"Plus d" condition

This control condition causes an A register entry of the form 0000XX, where the highest order 12 bits of the A register are cleared and the lowest order six bits correspond to the lowest order six bits of the X register. This condition is defined as follows:

Condition: (f) = 14

"Minus d" condition

This control condition causes an A register entry of the form 7777NN, where the highest order 12 bits of the A register are forced to ones and the lowest order six bits are set to the complement of the lowest order six bits of the X register content. This condition is defined as follows:

Condition: (f) = 15

"A + d" condition

This control condition causes an A register entry of the ones complement sum of (A) + 0000XX, where the highest order 12 bits of the storage operand are zero and the lowest order six bits correspond to the lowest order six bits of the X register content. This condition is defined as follows:

Condition: (f) = 16

"A - d" condition

This control condition causes an A register entry of the ones complement sum of (A) + 7777NN, where the highest order 12 bits of the storage operand are forced one and the lowest order six bits correspond to the complement of the lowest order six bits in the X register. This condition is defined as follows:

Condition: (f) = 17

"A .LD. d" condition

This control condition causes an A register entry of the bit by bit logical difference of (A) and 0000XX, where the highest order 12 bits of the storage operand are zero and the lowest order six bits correspond to the lowest order six bits in the X register. The logical difference of two binary quantities A and B is defined to be:

A .and..not. B .or. B .and..not. A

This control condition is defined as follows:

Condition: (f) = 11

"A .LP. d" condition

This control condition causes an A register entry of the bit by bit logical product of (A) and 0000XX, where the highest order 12 bits of the storage operand are zero and the lowest order six bits correspond to the lowest order six bits in the X register. This condition is defined as follows:

Condition: (f) = 12

"A .SC. d" condition

This control condition causes an A register entry of the bit by bit logical product of (A) and 7777NN, where the highest order 12 bits of the storage operand are forced one and the lowest order six bits are the complement of the lowest order six bits in the X register. This condition is defined as follows:

Condition: (f) = 13

"Plus X" condition

This control condition causes an A register entry of the form 00XXXX, where the highest order six bits of the A register are cleared and the lowest order 12 bits correspond to the 12 bits in the X register. This condition is defined as follows:

Condition: (f) = 30,40,50

"A + X" condition

This control condition causes an A register entry of the ones complement sum of (A) + 00XXXX, where the highest order six bits of the storage operand are zero and the lowest order 12 bits correspond to the 12 bits in the X register. This condition is defined as follows:

Condition: (f) = 31,41,51
or: (f) = 35,36,37 & (k) = 1
or: (f) = 45,46,47 & (k) = 2
or: (f) = 55,56,57 & (k) = 3

"A - X" condition

This control condition causes an A register entry of the ones complement sum of (A) + 77NNNN, where the highest order six bits of the storage operand are forced one and the lowest order 12 bits correspond to the complement of the bits in the X register. This condition is defined as follows:

Condition: (f) = 32,42,52

"A .LD. X" condition

This control condition causes an A register entry of the bit by bit logical difference of (A) and 00XXXX, where the highest order six bits of the storage operand are zero and the lowest order 12 bits correspond to the bits in the X register. The logical difference of two binary quantities A and B is defined to be:

A .and..not. B .or. B .and..not. A

This control condition is defined as follows:

Condition: (f) = 33,43,53

"Input channel" condition

This control condition causes an A register entry of the form OOCCCC, where the highest order six bits of the A register are cleared and the lowest order 12 bits correspond to the 12 bits currently on the selected input channel data path. The proper input channel is selected by the lowest order three bits in the d register. This condition is defined as follows:

Condition: (f) = 70

"Plus QX" condition

This control condition causes an A register entry of the form QQXXXX, where the highest order six bits of the A register are set to correspond with the lowest order six bits of the Q register, and the lowest order 12 bits of the A register are set to correspond with the bits in the X register. This condition is defined as follows:

Condition: (f) = 20

"A + QX" condition

This control condition causes an A register entry of the ones complement sum of (A) + QQXXXX, where the highest order six bits of the storage operand correspond to the lowest order six bits of the Q register, and the lowest order 12 bits of the storage operand correspond to the bits in the X register. This condition is defined as follows:

Condition: (f) = 21

"A .LP. QX" condition

This control condition causes an A register entry of the bit by bit logical product of (A) and QQXXXX, where the highest order six bits of the storage operand correspond to the lowest order six bits of the Q register, and the lowest order 12 bits of the storage operand correspond to the bits in the X register. This condition is defined as follows:

Condition: (f) = 22

"A .LD. QX" condition

This control condition causes an A register entry of the bit by bit logical difference of (A) and QQXXXX, where the highest order six bits of the storage operand correspond to the lowest order six bits of the Q register, and the lowest order 12 bits of the storage operand correspond to the bits in the X register. The logical difference of two binary quantities is defined to be:

A .and..not. B .or. B .and..not. A

This control condition is defined as follows:

Condition: (f) = 23

"Left shift" condition

This control condition causes an A register entry which shifts the previous contents of the A register left circularly by one bit position. The lowest order bit position of the A register is filled with the previous content of the highest order bit position. All other bit positions are filled with the previous content of the next lower bit position. This control condition is defined as follows:

Condition: lowest order five bits in (sk) $\neq$ 0 & highest order bit in (sk) = 0

"Right shift" condition

This control condition causes an A register entry which shifts the previous contents of the A register right open ended by one bit position. The highest order bit of the A register is set to zero in this process. All other bit positions are filled with the previous content of the next highest order bit position. This control condition is defined as follows:

Condition: lowest order five bits in (sk) $\neq$ 0 & highest order bit in (sk) = 1

Shift count register (sk)

The sk register is a six bit clear/enter type register with gated clock pulse control. This register is cleared, and six bits of new data are entered from the d register, at the end of a clock period in which the following condition exists.

Condition: (f) = 10 & T0F & no T1F

If the highest order bit in the d register is zero at the time of the sk register entry, the lowest order five bits of the sk register are entered with copies of the lowest order five bits of the d register. If the highest order bit in the d register is one at the time of the sk register entry, the lowest order five bits of the sk register are entered with the complement of the corresponding bits in the d register. In either case, the highest order bit in the sk register is entered with a copy of the highest order bit in the d register.

The lowest order five bits in the sk register contain the count of the number of bit positions by which the A register content is to be shifted. If the highest order bit in the sk register is zero, the shift is to the left in a circular mode. If the highest order bit in the sk register is one, the shift is to the right in an open ended mode. The shift of the A register begins in the clock period following entry of the sk register. The shift continues one bit position per clock period until the required number of shifts have occurred. The lowest order five bits of the sk register content are counted down as the shifting takes place. The lowest order five bits of the sk register are cleared, and a new five bit quantity one count less than the previous content is entered, at the end of a clock period in which the following condition exists. The highest order bit in the sk register is not altered in this process.

Condition: lowest order five bits in (sk) $\neq$ 0

"sk lockout" condition

This condition is a translation of the lowest order five bits in the sk register. The condition is present when the shift count represented by the lowest order five bits of (sk) has a value of four or more. The condition is formed by sensing bits two, three, and four in the sk register. This lockout condition prevents an enable issue condition from occurring until the shift count on a shift instruction has been reduced to a value of three or less.

Address Arithmetic

Address arithmetic in a PPU is computed in a 12 bit ones complement mode. There are two registers which perform major roles in the address computation. One register is the program address register (P), which provides storage for the current program address. The other is the operand address register (Q), which provides temporary storage during the computation of an operand address. Each of these registers is 12 bits in length. There are two static adder networks in the address arithmetic section. These networks and their interconnections are illustrated in figure 4-3 on the following page. The increment adder network is normally used for advancing the program address by one count as the instructions are executed. The address adder network is normally used to form the sum of a base address and the content of an index register for an operand storage reference. Each of the networks has alternate modes to accomplish special addressing functions. All addresses delivered to the storage address registers pass through these adders and result from the ones complement sum of two 12 bit quantities. As a result of this approach the last address in the core memory cannot be addressed. The address 7777 octal cannot be formed in the address arithmetic networks. There are 4095 decimal words of useful storage in the PPU.

Program address register (P)

The P register is a 12 bit clear/enter type register with gated clock pulse control. This register is cleared, and a new program address entered, as each instruction word in a program is executed. The P register contains the current program instruction address at all times in the execution of a program. Some instructions require a two word instruction format. In these cases the program address register content is advanced twice in the execution of the instruction. When a program branch occurs the P register is cleared and the initial address for the next program sequence is entered. The quantity entered in the P register is defined by the control conditions listed under the increment adder network and address adder network descriptions. The P register is cleared, and new data is entered, at the end of a clock period in which the following condition exists:

Condition: (f) = 00 thru 27, 50 thru 77 & (k) = 0 & GRF
or: (f) = 01,02 & (k) = 2,3 & GRF
or: (f) = 20 thru 27,30,31,32,33,34,60 thru 67 & (k) = 1 & GRF
or: (f) = 35,36,37,40,41,42,43,44 & (k) = 2 & GRF
or: (f) = 45,46,47,50,51,52,53,54,71 & (k) = 3 & GRF
or: (f) = 55,56,57 & (k) = 4 & GRF
or: (f) = 71,73 & (k) = 1,2 & "A zero" & GRF

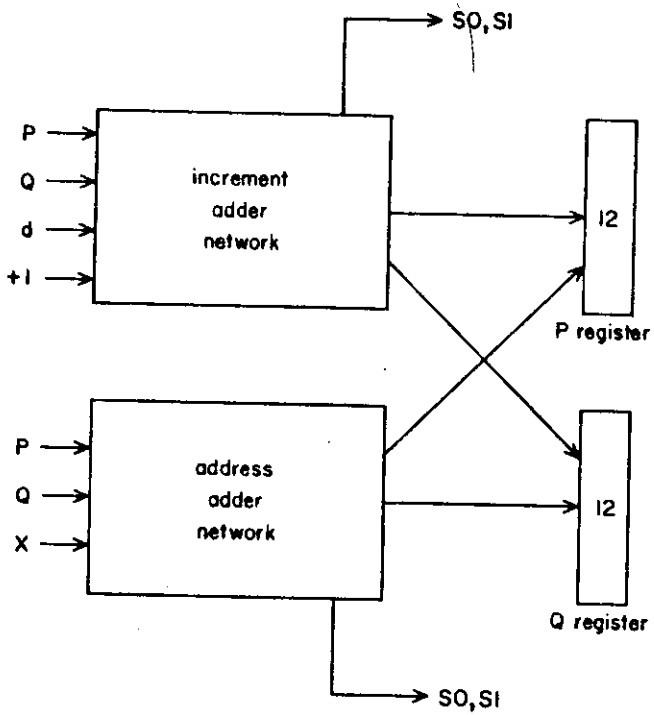


Fig. 4-3 Address Arithmetic

Operand address register (Q)

The Q register is a 12 bit clear/enter type register with gated clock pulse control. This register is used primarily for holding the address for an operand during the execution of an instruction. This register does not provide continuity for data between instructions. The Q register is also used to assist the X register in operand arithmetic where an 18 bit storage operand is required. This function is described in the section on operand arithmetic. The content of the Q register is cleared, and new data is entered, at the end of a clock period in which the following condition exists. The quantity entered in the Q register is defined by the control conditions listed under the increment adder network and address adder network descriptions.

Condition: (f) = 00 thru 77 & (k) = 0 & GRF
or: (f) = 00 thru 34, 40 thru 77 & (k) = 1 & GRF
or: (f) = 00 thru 44, 50 thru 77 & (k) = 2 & GRF
or: (f) = 00 thru 54, 60 thru 77 & (k) = 3 & GRF
or: (f) = 00 thru 77 & (k) = 4 & GRF

Increment adder network

The increment adder is a static network which forms the sum of two 12 bit numbers in a 12 bit ones complement mode. This network is normally used to advance the content of the P register by one count in the execution of a program sequence. It is also used to advance the content of the Q register by one count in the execution of block input or block output instructions where the Q register holds the operand address. An alternative mode for the increment adder is used to increment the program address by (d) in the execution of the relative jump instructions. The various control conditions which select the increment adder mode are listed below.

"P + 1 to P" condition

This control condition selects an increment adder mode which forms the ones complement sum of (P) + 0001 and delivers this result to the P register. This mode is used to advance the program address in the normal execution of a program sequence. The control condition is defined as follows:

```
Condition:  (f) = 00,01,02 & (k) = 0
      or:   (f) = 02 & (k) = 2
      or:   (f) = 04 & no "A zero"
      or:   (f) = 05 & "A zero"
      or:   (f) = 06 & "A negative"
      or:   (f) = 07 & no "A negative"
      or:   (f) = 10 thru 34,40,41,42,43,44,70,72,74,75,76,77
      or:   (f) = 35,36,37 & (k) = 0,2
      or:   (f) = 45,46,47,71,73 & (k) = 0,1,3
      or:   (f) = 50,51,52,53,54 & (k) = 0,2,3
      or:   (f) = 55,56,57 & (k) = 0,1,2,4
      or:   (f) = 60 thru 67 & no JDF
      or:   (f) = 71,73 & (k) = 2 & "A zero"
```

"P + d to P" condition

This control condition selects an increment adder mode which forms the ones complement sum of (P) + xxdd and delivers this result to the P register. The storage operand in this case is formed by extending the highest order bit in the d register to make a 12 bit operand. The lowest order six bits are direct copies of the six bits in the d register. The highest order six bits are zeros if the highest order bit in the d register is zero. The highest order six bits are ones if the highest order bit in the d register is a one. This mode of operation has the effect of incrementing, or decrementing, the content of the P register by the quantity in the d register. In this process the content of the d register is considered as a signed integer. The control condition that selects this mode is defined as follows:

```
Condition:  (f) = 03
      or:   (f) = 04 & "A zero"
      or:   (f) = 05 & no "A zero"
      or:   (f) = 06 & no "A negative"
      or:   (f) = 07 & "A negative"
```

"P + 1 to Q" condition

This control condition selects an increment adder mode which forms the ones complement sum of (P) + 0001 and delivers this result to the Q register. This control condition is defined as follows:

```
Condition:  (f) = 01 & (k) = 2
      or:   (f) = 60 thru 67 & JDF
```

"Q + 1 to Q" condition

This control condition selects an increment adder mode which forms the ones complement sum of (Q) + 0001 and delivers this result to the Q register. This mode is used to advance the operand address in block copy instructions. The condition is defined as follows:

Condition: (f) = 71,73 & (k) = 2 & no "A zero"

"Q + 1 to P" condition

This control condition selects an increment adder mode which forms the ones complement sum of (Q) + 0001 and delivers this result to the P register. This mode is used in the return jump instruction to enter the P register with the new program sequence address. The condition is defined as follows:

Condition: (f) = 02 & (k) = 3

"P + 1 to S" condition

This control condition selects an increment adder mode which forms the ones complement sum of (P) + 0001 and delivers this result to the S registers. This condition overlaps the control condition "P + 1 to P". The increment adder network delivers the same result to the P register and to the S registers in this case. This condition is defined as follows:

Condition: (f) = 00,01,02,10 thru 27,50 thru 77 & (k) = 0
or: (f) = 04 & no "A zero"
or: (f) = 05 & "A zero"
or: (f) = 06 & "A negative"
or: (f) = 07 & no "A negative"

"P + d to S" condition

This control condition selects an increment adder mode which forms the ones complement sum of (P) + xxdd and delivers this result to the S registers. The storage operand in this case is formed by extending the highest order bit in the d register to make a 12 bit operand. This condition is identical to the control condition "P + d to P". There are two destinations for the increment adder results in this case.

"Q to S" condition

This control condition selects an increment adder mode which forms the ones complement sum of (Q) + 0000 and delivers this result to the S registers. Note that this is not the same as transmitting the content of the Q register to the S registers. The result of adding 7777 + 0000 is all zeros in a ones complement mode. This condition is defined as follows:

Condition: (f) = 01,02,35,36,37 & (k) = 1
or: (f) = 45,46,47 & (k) = 2
or: (f) = 55,56,57 & (k) = 3
or: (f) = 50,51,52,53,54,55,56,57 & (k) = 1

"Q + 1 to S" condition

This control condition selects an increment adder mode which forms the ones complement sum of (Q) + 0001 and delivers this result to the S registers. This condition overlaps the control condition "Q + 1 to Q". In these cases the increment adder network delivers the same result to both the Q register and the S registers. This condition is defined as follows:

Condition: (f) = 02 & (k) = 3
or: (f) = 71,73 & (k) = 2 & no "A zero"

Address adder network

The address adder is a static network which forms the sum of two 12 bit numbers in a 12 bit ones complement mode. This network is normally used to form an operand address from a base address and an index register content. The result of this operation may be transmitted to the P register, the Q register, or the S registers. The various control conditions which select the address adder mode are listed below.

"X + Q to Q" condition

This control condition selects an address adder mode which forms the ones complement sum of (X) + (Q) and delivers this result to the Q register. This is the normal mode for adding an index register content to a base address. The condition is defined as follows:

Condition: (f) = 02,50,51,52,53,54,55,56,57 & (k) = 2

"X to Q" condition

This control condition selects an address adder mode which forms the ones complement sum of (X) + 0000 and delivers this result to the Q register. Note that this is not the same as transmitting the content of the X register to the Q register. The result of adding 7777 + 0000 is all zeros in a ones complement mode. This condition is defined as follows:

Condition: (f) = 00 thru 77 & (k) = 1,3,4
or: (f) = 00 thru 47, 60 thru 77 & (k) = 2

"d to Q" condition

This control condition selects an address adder mode which forms the ones complement sum of 00XX + 0000 and delivers this result to the Q register. The storage operand in this case has zeros in the upper six bit positions. The lowest order six bits correspond to the lowest order six bits in the X register. This operand is formed by blocking the highest order six bits in the data path from the X register to the address adder. The result is the same as if the contents of the d register were transmitted to the address adder without extension. This mode is used to address one of the 64 index registers in the first phase of instruction execution. The condition is defined as follows:

Condition: (f) = 00 thru 77 & (k) = 0

"X to P" condition

This control condition selects an address adder mode which forms the ones complement sum of (X) + 0000 and delivers this result to the P register. Note that this is not the same as transmitting the content of the X register to the P register. The result of adding 7777 + 0000 is all zeros in a ones complement mode. This condition is defined as follows:

Condition: (f) = 60 thru 67 & (k) = 1 & JDF

"X + Q to P" condition

This control condition selects an address adder mode which forms the ones complement sum of (X) + (Q) and delivers this result to the P register. This condition is defined as follows:

Condition: (f) = 01 & (k) = 2

"X + Q to S" condition

This control condition selects an address adder mode which forms the ones complement sum of (X) + (Q) and delivers this result to the S registers. This condition overlaps the conditions which transmit the same result to the P register or the Q register. This condition is defined as follows:

Condition: (f) = 01,02,50,51,52,53,54,55,56,57 & (k) = 2

"X to S" condition

This control condition selects an address adder mode which forms the ones complement sum of (X) + 0000 and delivers this result to the S registers. Note that this is not the same as transmitting the content of the X register to the S registers. The result of adding 7777 + 0000 is all zeros in a ones complement mode. This condition is defined as follows:

Condition: (f) = 40 thru 47 & (k) = 1
or: (f) = 60 thru 67 & (k) = 1 & JDF
or: (f) = 71,73 & (k) = 1 & no "A zero"

"d to S" condition

This control condition selects an address adder mode which forms the ones complement sum of 00XX + 0000 and delivers this result to the S registers. The storage operand in this case has zeros in the upper six bit positions. The lowest order six bits correspond to the lowest order six bits in the X register. This operand is formed by blocking the highest order six bits in the data path from the X register to the address adder. The result is the same as if the contents of the d register were transmitted to the address adder without extension. This mode is used to address one of the 64 index registers in the first phase of instruction execution. The condition is defined as follows:

Condition: (f) = 30 thru 47 & (k) = 0

Internal Storage

The internal storage for a PPU consists of two independent banks of magnetic core memory. Each bank contains 2048 words 12 bits in length. A bank has a read access time of four clock periods and a read/write cycle time of ten clock periods. The PPU storage banks contain the same storage modules as are used in SCM for the CPU. The organization of a PPU storage bank is illustrated in figure 4-4 on the following page.

The following description for a PPU storage bank applies to each of the two banks in the PPU internal storage. Where the two banks must be distinguished they are called "bank 0" and "bank 1." The S register and Z register illustrated in figure 4-4 are duplicated in the two banks. Where the S registers are referenced individually they are called the S0 and S1 registers. In a corresponding manner the Z registers are called Z0 and Z1. The X register illustrated in figure 4-4 is not duplicated. It appears in this figure to illustrate the sense amplifier data destination. The data from the bank 0 sense amplifier merges with the data from the bank 1 sense amplifier in entering the X register.

The two banks in a PPU storage section are connected so that consecutive addresses alternate between the banks. This is accomplished by using the lowest order bit in the 12 bit storage address as the bank selection bit. The highest order 11 bits are transmitted to the individual bank S registers. Within a PPU storage bank the storage modules are connected so that consecutive addresses within the bank alternate between the two storage modules. This is accomplished by using the lowest order bit in the 11 bit bank address as the module selection bit. The result of these selection bit locations is that consecutive program addresses reference all four storage modules before returning to the first module for a second reference.

The parity generation network illustrated in figure 4-4 is common to the two banks. This network has a 13 bit output path to each of the two Z registers. Each bank has its own storage sequence control. The enable issue condition is transmitted from the PPU control section to each of the two bank sequence controls. A bank sequence control is able to act on the enable issue condition only when the lowest order bit in the storage address has selected that bank. Data is transmitted directly from the sense amplifiers to the f and d registers at the end of those storage read periods in which an instruction is read from storage. The control condition for this transmission is defined in the descriptions of those registers.

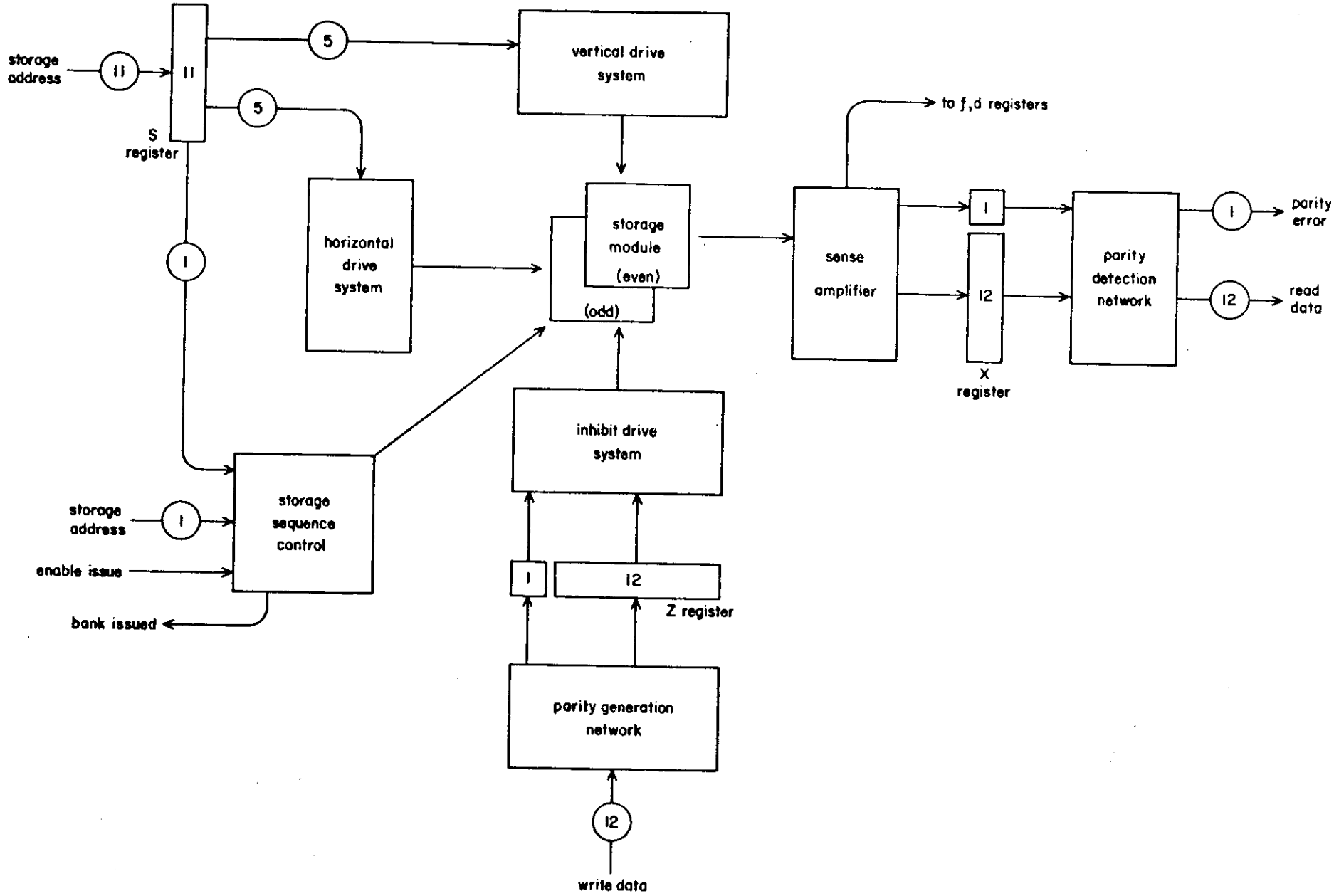


Fig 4-4 PDU Storage Bank

Bank sequence control

Each PPU storage bank has a bank sequence control unit. This unit forms a sequence of ten control conditions which govern the activity within the storage bank in ten consecutive clock periods. The sequence begins at the end of a clock period in which the lowest order bit in the storage address from the address arithmetic section selects the subject bank, an "enable issue" condition is present, and the subject bank is not busy. The sequence is chronologically listed below.

Transmit address to S register
Enable issue condition
No bank busy flag
Set bank busy flag

Begin read drive current
Transmit bank issued condition to instruction control

CP02

CP03

End read drive current
Transmit data from sense amplifier to X register

Transmit write data to Z register

Begin write drive current
Begin selected inhibit drive currents

CP07

CP08

CP09 End write drive current
End selected inhibit drive currents
Clear bank busy flag

Bank busy flag

This flag is a part of the bank sequence control unit and is the mechanism to lock out further initiations of a bank read/write cycle until a previously initiated cycle is completed. This flag is a bit register with a separate set and a separate clear input. The flag is set and cleared by the storage sequence control unit as indicated in the timing chart above. It is set for a total of nine clock periods.

S register

Each PPU storage bank contains an 11 bit S register. This is a clear/enter type register with gated clock pulse control. The content of the S register is cleared, and new data is entered from the address arithmetic section of the PPU, at the end of a clock period in which the following condition is present:

Condition: no bank busy flag

The condition defined above causes the S register to clear and enter data from the address arithmetic section at the end of every clock period in which the storage bank is idle. The only data which is significant to the operation of the storage bank is the data transmitted in the last clock period before a storage read/write cycle begins. This address is then held in the S register for the ten clock periods of the storage cycle.

PPU storage modules

Each PPU storage bank contains two storage modules. These modules are the same type as are used in SCM for the CPU. Each module contains a core memory stack containing 32 x 32 x 13 cores plus the associated supporting circuits. These modules each supply 1024 twelve bit words plus a parity bit for each 12 bit word. All words with an even numbered bank address are stored in one module, and all words with an odd numbered bank address are stored in the other module. The horizontal and vertical drive system for the storage selection is duplicated in the two storage modules. Bits six through ten of the address in the S register are translated into the horizontal drive selection. Bits one through five are translated into the vertical drive selection. Bit zero of the S register is not used in the storage modules. This bit is used by the storage sequence control unit to select between the two storage modules.

Parity generation network

The parity generation network illustrated in figure 4-4 is common to the two storage banks in a PPU. This network receives 12 bits of data from the write data selection unit during every clock period. The parity generation network forms a 13th bit with a value such that an odd number of bits in the resulting 13 bit word will have a value of one. This 13 bit word is then delivered to both Z registers.

Z register

Each PPU storage bank contains a 12 bit Z register plus a parity bit register. This is in effect a 13 bit clear/enter type register with gated clock pulse control. The Z register is cleared, and new data is entered from the parity generation network, at the end of CP05 in the bank sequence timing chain. This data is held in the Z register for four clock periods. The data is cleared, and a word of all zeros is entered, at the end of CP09 in the bank sequence timing chain. The data in the Z register directly controls the 13 inhibit drive circuits for the storage modules.

X register

The X register is a 12 bit clear/enter type register with gated clock pulse control. This register is the storage data readout register and is common to the two PPU storage banks. A one bit parity register is associated with the X register. This bit may be considered a 13th bit of the X register in that the control for this bit is the same as the control for the 12 bits of significant data. The data in the X register is cleared, and new data is entered from the storage bank sense amplifiers, at the end of CP04 in the bank sequence timing chain. This data then remains in the X register until the next time CP04 in one of the storage bank timing chains. Only one of the two PPU storage banks can begin a read/write cycle in a given clock period. As a result only one bank can cause an X register entry in a given clock period. The minimum time from the entry of data in the X register from one bank to the entry of data from the other bank is five clock periods.

In addition to the normal operation of the X register described above, the X register content is statically cleared to zero during a PPU "dead start" condition.

Parity detection network

The parity detection network illustrated in figure 4-4 continually monitors the data in the X register. This network generates a parity error condition during any clock period in which the 13 bit X register content does not have an odd number of one bits. The parity error condition then causes an error flag to set in the parity error register for the PPU.

Write Data Selection

Each read/write cycle in the magnetic core storage unit destroys the information in the addressed word position of the selected storage module. New information is written into this storage location via the appropriate Z register. The original information may be restored by transmitting the data from the X register to the Z register during CP05 of the storage bank timing sequence. This is the normal case in reading data from storage. There are three alternate paths for writing data into storage from other parts of the PPU. These data paths are selected by static networks as illustrated in figure 4-5 on the following page.

The channel selection network illustrated in figure 4-5 is the same network used in the operand arithmetic unit. This network selects 12 bits of input data from one of the eight input channels. The data is then transmitted to both the operand arithmetic mode selection network and to the write data mode selection network.

The write data mode selection network illustrated in figure 4-5 chooses one of four possible data paths for transmission of data to the Z registers. These data paths are all 12 bits wide. The data path from the A register transmits only the lowest order 12 bits in that register. The selection modes are controlled by two write data mode flags.

Write data mode flags

The two write data mode flags illustrated in figure 4-5 are clear/enter type bit registers with gated clock pulse control. These two flags select one of the four data paths in the mode selection network. These flags are cleared, and new data is entered, at the end of a clock period in which the following condition exists:

Condition: GRF

The information entered in the write data mode flags is determined by the static conditions defined below. These conditions are encoded into the two flags and then translated into the four conditions again for mode selection.

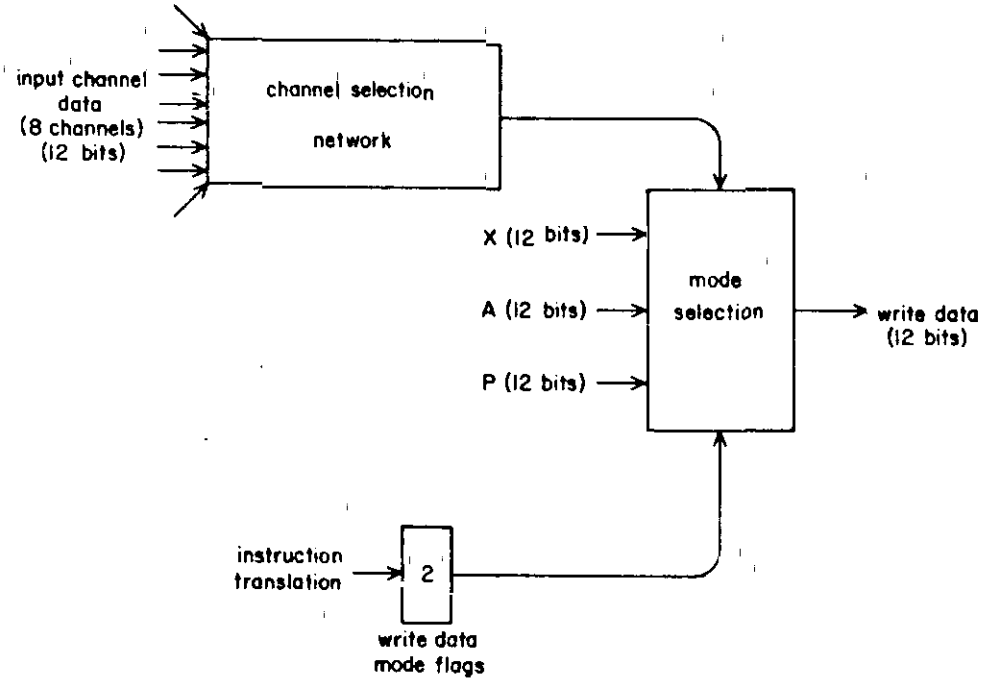


Fig. 4-5 Write Data Selection

"P to Z" condition

This control condition causes the mode selection network to transmit the content of the P register to the Z registers. This condition is defined as follows:

Condition: (f) = 02 & (k) = 2

"A to Z" condition

This control condition causes the mode selection network to transmit the lowest order 12 bits in the A register to the Z registers. This condition is defined as follows:

Condition: (f) = 34 & (k) = 0
or: (f) = 35,36,37,44 & (k) = 1
or: (f) = 45,46,47,54 & (k) = 2
or: (f) = 55,56,57 & (k) = 3

"Input to Z" condition

This control condition causes the mode selection network to transmit the selected input channel data to the Z registers. This condition is defined as follows:

Condition: (f) = 71 & (k) = 1,2 & no "A zero"

"X to Z" condition

This control condition causes the mode selection network to transmit the content of the X register to the Z registers. This is the normal storage restoration path and is defined as follows:

Not: "P to Z"
nor: "A to Z"
nor: "input to Z"

Instruction Translation

The instruction translation portion of a PPU consists of three registers and a static translation network as illustrated in figure 4-6 on the following page. The f register and d register are loaded with information at the beginning of instruction execution. These registers are not altered during the execution of an instruction. The k register is cleared at the beginning of instruction execution. The content of the k register is advanced as the instruction requires additional storage references.

f register

The f register is a six bit clear/enter type register with gated clock pulse control. This register holds the highest order six bits in an instruction word throughout the execution of the instruction. The data entering the f register comes directly from the storage sense amplifiers and is gated into the f register at the end of the same clock period that the data is entered in the X register. Only the highest order six bits of the 12 data bits read from storage are entered in the f register, and only during those storage read/write cycles in which (k) = 0. The f register data is cleared, and new data is entered, at the end of a clock period in which the following condition is present:

Condition: CP04 in bank 0 & (k) = 0
or: CP04 in bank 1 & (k) = 0

In addition to the normal operation of the f register described above, there are two static conditions which force the f register content to special values. These conditions occur only during the initial dead start of a PPU. The f register is forced to an octal value of 71 during the "dead start" condition. In addition, the second bit of the f register is set during a "dead dump" condition. This causes an f register value of 73 when a "dead start" is followed by a "dead dump."

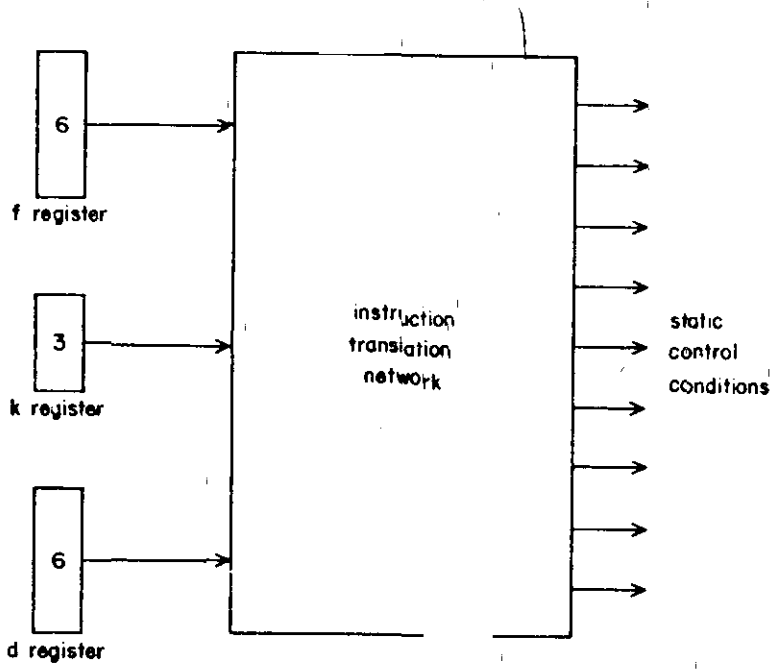


Fig. 4-6 Instruction Translation

k register

The k register is a three bit clear/enter type register with gated clock pulse control. This register contains an instruction phase indicator which alters the static control conditions as each storage reference is made during the execution of an instruction. The k register content is forced to an octal value of one during a "dead start" condition. The content of the k register is cleared, and new data entered, at the end of a clock period in which the following condition exists:

Condition: GRF

The data entered in the k register is formed directly in the instruction translation network. The octal value is listed below for each set of control conditions.

Set (k) to zero

The k register input data has a zero value during those clock periods in which the following condition is present:

Condition: (f) = 00,03 thru 17,24 thru 27,70,72,74,75,76,77 & (k) = 0
or: (f) = 20,21,22,23,30,31,32,33,34,60 thru 67 & (k) = 1
or: (f) = 71,73 & (k) = 1,2 & "A zero"
or: (f) = 01,35,36,37,40,41,42,43,44 & (k) = 2
or: (f) = 02,45,46,47,50,51,52,53,54,71 & (k) = 3
or: (f) = 55,56,57 & (k) = 4

Set (k) to one

The k register input data has an octal value of one during those clock periods in which the following condition is present:

Condition: (f) = 01,02,50 thru 57 & (k) = 0 & (d) $\neq$ 0
or: (f) = 20,21,22,23,30 thru 47,60 thru 67,71,73 & (k) = 0

Set (k) to two

The k register input data has an octal value of two during those clock periods in which the following condition is present:

Condition: (f) = 01,02,50 thru 57 & (k) = 0 & (d) = 0
or: (f) = 01,02,35,36,37,40 thru 57 & (k) = 1
or: (f) = 71,73 & (k) = 1 & no "A zero"
or: (f) = 73 & (k) = 2 & no "A zero"
or: (f) = 71 & (k) = 2 & no IRF & no "A zero"

Set (k) to three

The k register input data has an octal value of three during those clock periods in which the following condition is present:

Condition: (f) = 02,45,46,47,50 thru 57 & (k) = 2
or: (f) = 71 & (k) = 2 & IRF

Set (k) to four

The k register input data has an octal value of four during those clock periods in which the following condition is present:

Condition: (f) = 55,56,57 & (k) = 3

d register

The d register is a six bit clear/enter type register with gated clock pulse control. This register holds the lowest order six bits in an instruction word throughout the execution of the instruction. The data entered in the d register comes directly from the storage sense amplifiers and is gated into the d register at the end of the same clock period that the data is entered in the X register. Only the lowest order six bits of the 12 data bits read from storage are entered in the d register, and only during those storage read/write cycles in which (k) = 0. The d register data is cleared, and new data is entered, at the end of a clock period in which the following condition is present:

Condition: CP04 in bank 0 & (k) = 0
or: CP04 in bank 1 & (k) = 0

The d register content is forced to zero during a "dead start" condition. This condition occurs only during the initial start of a PPU. The d register content is used during the execution of some instructions as an arithmetic operand. In these cases the data is actually taken from the lowest order six bits in the X register. The shift instruction uses the data in the d register as a shift count. In this case the data is transmitted from the d register to the sk register for processing. The lowest order three bits of the d register content are statically translated to control the input channel and output channel selection networks.

Instruction translation network

Most of the static control conditions generated in the instruction translation network are defined elsewhere in this part of the reference manual as a part of the description of the device they control. Almost all of these control conditions contain translations of the f register and the k register. The instruction translation network illustrated in figure 4-6 is therefore somewhat distributed throughout the PPU. The following control conditions are not defined elsewhere.

"Enable issue" condition

This control condition enables a storage bank sequence control to begin a read/write cycle when the selected bank is free. This condition is defined as follows:

```
Not:  "A jump" & no T3F
nor:  "sk lockout"
nor:  (f) = 00,77
nor:  (f) = 71 & (k) = 1 & no "A zero" & no IWF
nor:  (f) = 71 & (k) = 2 & no "A zero" & no IWF & no IRF
nor:  (f) = 70 & no IWF
nor:  (f) = 72 & OWF
nor:  (f) = 73 & (k) = 2 & OWF
nor:  (f) = 70,72 & no T4F
nor:  (f) = 10 & no T1F
nor:  no T0F
```

"A jump" condition

This control condition groups the five instruction translations which cause a branch in the program sequence relative to the current program address. The condition is present when the jump is taken.

```
Condition:  (f) = 03
      or:   (f) = 04 & "A zero"
      or:   (f) = 05 & no "A zero"
      or:   (f) = 06 & no "A negative"
      or:   (f) = 07 & "A negative"
```

"A negative" condition

This control condition is a direct copy of the highest order bit in the A register. The condition is present when the bit is one.

"A zero" condition

This control condition is a translation of all 18 bits in the A register. The condition is present when all 18 bits in the A register have a value of zero. Note that the condition of all ones is not considered as zero in this sense.

Instruction Timing

The timing of instruction execution in a PPU is closely tied to the storage references. The access time for instructions and operands is so much longer than the arithmetic operations involved that the storage access completely dominates the overall timing considerations. There are two clock periods in the storage read/write sequence which interface with the control section of the PPU directly. The transfer of control from arithmetic to storage occurs at the beginning of a read/write cycle in the form of the "enable issue" condition. The storage sequence control unit then begins the read/write cycle when the appropriate bank is free. The transfer of control from storage to arithmetic occurs at CP04 in the timing sequence. At the end of this clock period the data is read from storage to the X register and is available for processing. The interaction between the instruction timing chains and the storage timing chains is illustrated in figure 4-7 on the following page.

Go registers flag (GRF)

The GRF is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set for one clock period when the "enable issue" condition has been acted upon by a bank sequence control unit and a storage read/write cycle is in process. This flag is set to indicate to the operating registers in the PPU that the previous sequence of arithmetic operations is complete and a new operand is in the process of being fetched. At the end of this clock period the arithmetic results are entered in the operating registers and a new cycle of control begins. The GRF is set at the end of a clock period in which the following condition is present:

Condition: CP01 in bank 0
or: CP01 in bank 1

Main timing chain

The main timing chain is illustrated in figure 4-7 and consists of five control flags. These flags enter in the instruction translations to determine the timing of the various conditions. The chain begins a sequence as soon as data is read from storage to the X register. The various flags then provide the required delays for the arithmetic functions.

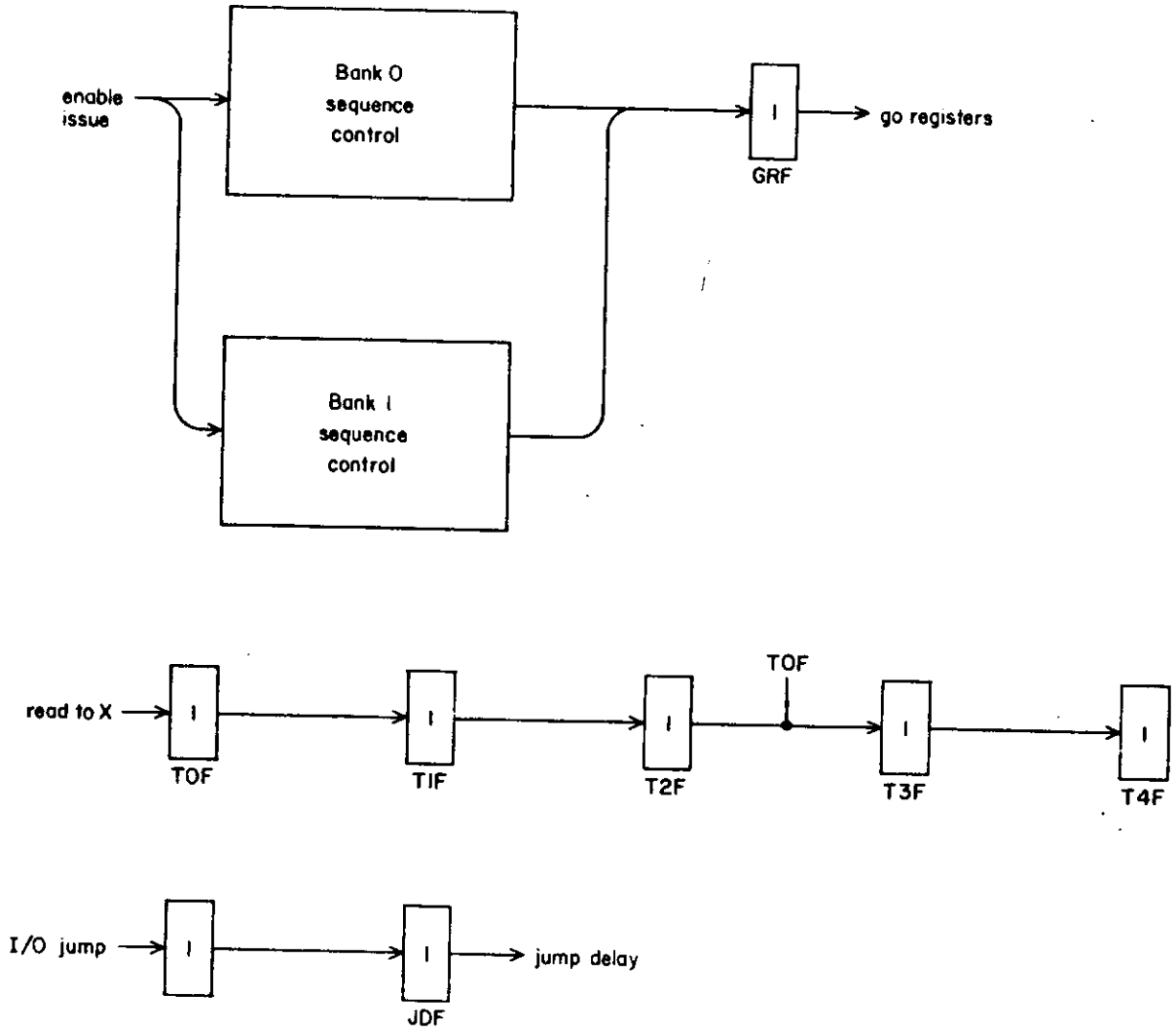


Fig. 4-7 Instruction Timing

Time 0 flag (T0F)

The T0F is a bit register with a separate set and a separate clear input. The T0F is set concurrently with the entry of data in the X register from storage. The T0F remains set until the arithmetic operations are completed and the next storage read/write cycle begins. It is cleared at the end of the same clock period at which the GRF is set. The T0F may be set for many clock periods, or for only two clock periods, depending on the time required to process the data from a storage reference.

The T0F is set at the end of a clock period in which the following condition exists:

Condition: CP04 in bank 0
or: CP04 in bank 1

The T0F is cleared at the end of a clock period in which the following condition exists:

Condition: CP01 in bank 0
or: CP01 in bank 1

In addition to the normal operation of the T0F described above, the T0F is forced set statically by a "dead start" condition. This condition occurs only at the time of dead starting the PPU.

Time 1 flag (T1F)

The T1F is a clear/enter type bit register which is cleared at the end of every clock period. This is the second flag in the main PPU timing chain. The T1F is set at the end of a clock period in which the T0F is set. The result is that the T1F appears to set one clock period later than the T0F and to remain set until one clock period after the T0F clears.

Time 2 flag (T2F)

The T2F is a clear/enter type bit register which is cleared at the end of every clock period. The T2F is set at the end of a clock period in which the T1F is set. The result is that the T2F appears to set one clock period later than the T1F and to remain set until one clock period after the T1F clears.

Time 3 flag (T3F)

The T3F is a clear/enter type bit register which is cleared at the end of every clock period. The T3F is set at the end of a clock period in which the T2F and the TOF are set. The result is that the T3F appears to set one clock period later than the T2F and to remain set until one clock period after the TOF clears.

Time 4 flag (T4F)

The T4F is a clear/enter type bit register which is cleared at the end of every clock period. The T4F is set at the end of a clock period in which the T3F is set. The result is that the T4F appears to set one clock period later than the T3F and to remain set until one clock period after the T3F clears.

Jump delay chain

A jump delay chain for I/O jump instructions is illustrated in figure 4-7. This chain consists of two control flags which determine the timing of instructions 60 through 67. The delay is necessary to cover the synchronizing time of the input and output channel control signals. The first flag in the jump delay chain is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set at the end of a clock period in which the following condition is present:

```
Condition:  (f) = 60 & (k) = 1 & IWF
      or:   (f) = 61 & (k) = 1 & no IWF
      or:   (f) = 62 & (k) = 1 & IRF
      or:   (f) = 63 & (k) = 1 & no IRF
      or:   (f) = 64 & (k) = 1 & OWF
      or:   (f) = 65 & (k) = 1 & no OWF
      or:   (f) = 66 & (k) = 1 & ORF
      or:   (f) = 67 & (k) = 1 & no ORF
```

Jump delay flag (JDF)

The JDF is a clear/enter type bit register which is cleared at the end of every clock period. The JDF is set at the end of a clock period in which the first flag in the jump delay chain is set. The JDF therefore appears to set one clock period after the first flag in the jump delay chain sets and remains set one clock period later.

Input Channels

There are provisions for eight input cables in each PPU. Each input cable provides 12 bits of incoming data and the associated control lines for that data. The PPU may sample the data on any one of these eight input cables at any one time. The selection of which one of the eight cables is determined by the lowest order three bits in the d register. The input channels are numbered zero through seven to correspond with the value of the lowest order three bits in the d register. The circuits which select the input data and input control lines are illustrated in figure 4-8 on the following page.

Input channel word flag

Each of the eight input data paths has associated with it an input channel word flag. This flag is a bit register with a separate set and a separate clear input. The flag is set when a word pulse is transmitted over the input cable to this PPU. The flag is cleared when the PPU has sampled the data on the cable and sends a resume pulse to the transmitting device at the other end of the cable. This flag is also forced to a cleared state during a "dead start" condition. The condition for clearing this flag may be defined as the end of a clock period in which the following condition exists:

Condition: (f) = 70 & GRF & (d) = channel number
or: (f) = 71 & (k) = 1 & no "A zero" & GRF & (d) = channel number
or: (f) = 71 & (k) = 2 & no "A zero" & no IRF & GRF & (d) = channel number
or: "dead start"

Input channel record flag

Each of the eight input data paths has associated with it an input channel record flag. This flag is a bit register with a separate set and a separate clear input. The flag is set when a record pulse is transmitted over the input cable to this PPU. The flag is cleared when the PPU has sampled the next following input data word and sends a resume pulse to the transmitting device at the other end of the cable. This flag is also forced to a cleared state during a "dead start" condition. The condition for clearing this flag is exactly the same as that for the input channel word flag defined above.

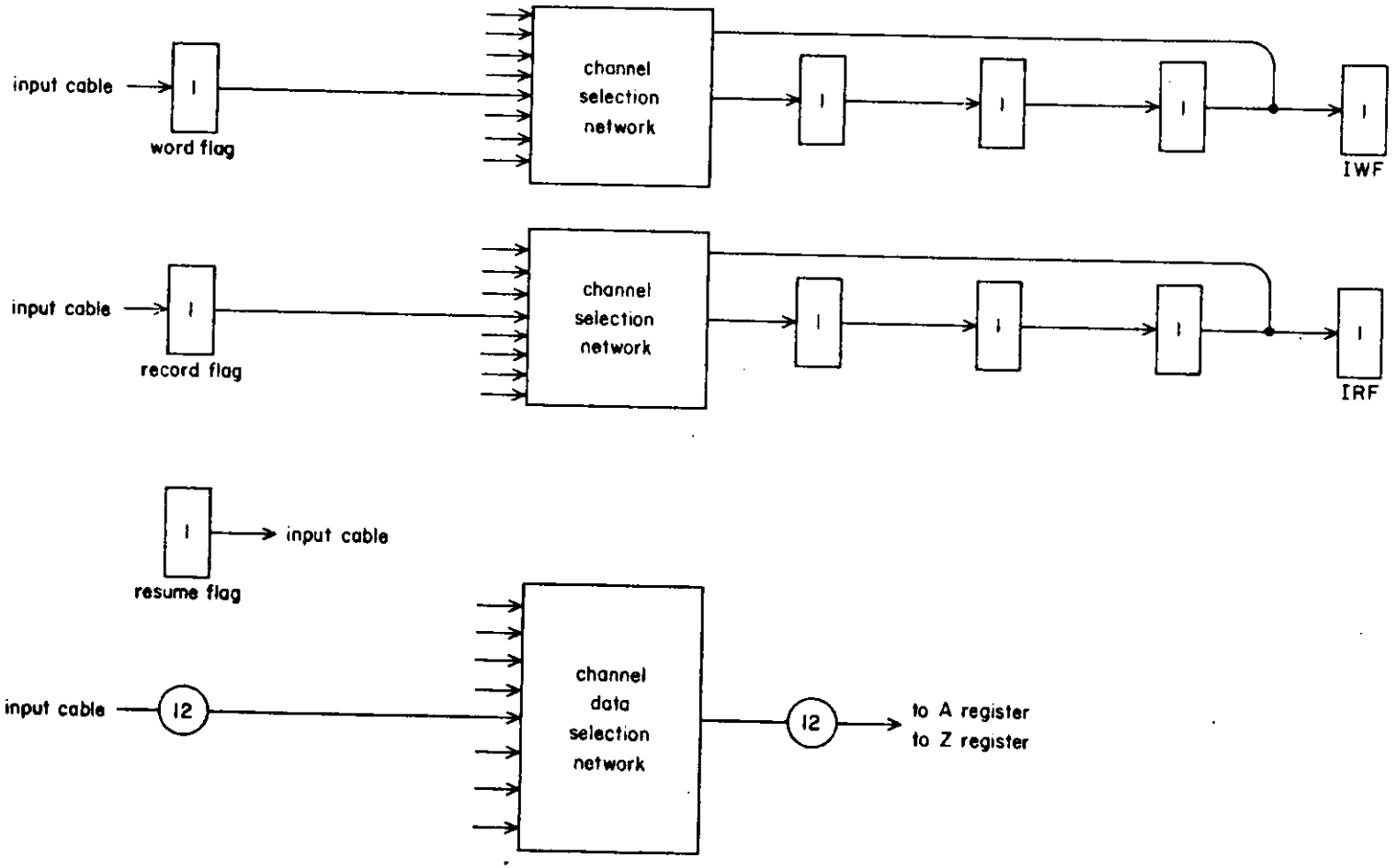


Fig. 4-8 Input Channels

Input channel resume flag

Each of the eight input data paths has associated with it an input channel resume flag. This flag is a clear/enter type bit register which is cleared at the end of every clock period. The flag is set for one clock period when the PPU has sampled the input data and is ready for the next word to be transmitted. This flag is also set during a "dead start" condition. A resume pulse is transmitted from this PPU over the input cable during the time in which this flag is set. This flag is set at the end of a clock period in which the following condition exists:

Condition: (f) = 70 & GRF & (d) = channel number
or: (f) = 71 & (k) = 1 & no "A zero" & GRF & (d) = channel number
or: (f) = 71 & (k) = 2 & no "A zero" & no IRF & GRF & (d) = channel number
or: "dead start"

IWF synchronizing network

Figure 4-8 illustrates a selection and synchronizing network which connects the eight individual input channel word flags to a common input word flag (IWF). A channel selection network receives the data from the eight input channel word flags. This network selects one flag on the basis of the value in the lowest order three bits in the d register. The selected flag is then transmitted through a series of three synchronizing flags to the IWF. This synchronizing chain is necessary because the individual channel flags are set asynchronous with the PPU clock. The chain provides the required time to resolve partial pulses resulting at the timing interface. Each of the synchronizing flags is a clear/enter type bit register which is cleared at the end of every clock period. Each is set at the end of a clock period in which the preceding flag is set. The result is that the IWF is set four clock periods after the selected input channel word flag is set, or four clock periods after the d register selects a new channel, whichever is later.

IRF synchronizing network

Figure 4-8 illustrates a selection and synchronizing network which connects the eight individual input channel record flags to a common input record flag (IRF). This network is identical to the IWF synchronizing network and performs the same function for the input record flags that the IWF synchronizing network performs for the input word flags.

Input word flag (IWF)

The input word flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set by the IWF synchronizing network and presents a common control signal to the PPU for all input channel word flags. The IWF represents a synchronized copy of the currently selected input channel word flag. The IWF is set at the end of a clock period in which the selected input channel word flag is set and the last flag in the IWF synchronizing chain is also set. The effect of this combination of conditions is that the IWF sets four clock periods after the input channel word flag sets, and clears one clock period after the input channel word flag clears. A new channel selection through the d register requires four clock periods for a correct interpretation of the newly selected input channel word flag.

Input record flag (IRF)

The input record flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set by the IRF synchronizing network and presents a common control signal to the PPU for all input channel record flags. The IRF represents a synchronized copy of the currently selected input channel record flag. The IRF is set at the end of a clock period in which the selected input channel record flag is set and the last flag in the IRF synchronizing chain is also set. The effect of this combination of conditions is that the IRF sets four clock periods after the input channel record flag sets, and clears one clock period after the input channel record flag clears. A new channel selection through the d register requires four clock periods for a correct interpretation o. the newly selected input channel record flag.

Channel data selection network

The channel data selection network illustrated in figure 4-8 selects 12 data bits from an input cable on the basis of the lowest order three bits in the d register. This is a static selection process which selects one word of 12 bits out of a possible eight groups of 12 bit words. The selected 12 bit word is transmitted to the operand arithmetic mode selection network illustrated in figure 4-2 and to the write data mode selection network illustrated in figure 4-5. Note that there is no input data register involved in this process. The data must be held at the transmitting device until the PPU has acknowledged the receipt of the data.

Output Channels

There are provisions for eight output cables in each PPU. Each output cable provides a path for 12 bits of outgoing data plus the associated control lines for that data. The PPU may enter data on any one of these eight output cables at any one time. The selection of which of the eight cables is determined by the lowest order three bits in the d register. The output channels are numbered zero through seven to correspond with the value in the lowest order three bits in the d register. The circuits which select the output data path and output control lines are illustrated in figure 4-9 on the following page.

Output channel word flag

Each of the eight output data paths has associated with it an output channel word flag. This flag is a bit register with a separate set and a separate clear input. The flag is set when a word pulse is transmitted over the associated output cable. The flag is cleared when a resume pulse is returned over this output cable. This flag is also forced to a cleared position during a "dead start" condition. A one clock period wide word pulse is formed for transmission over an output cable by a clear/enter type bit register which is cleared at the end of every clock period. This bit register is associated with the output channel word flag and sets the word flag in addition to transmitting the pulse over the output cable. This bit register is set at the end of a clock period in which the following condition exists:

Condition: (f) = 72 & GRF & (d) = channel number
or: (f) = 73 & (k) = 2 & GRF & (d) = channel number

Output channel record flag

Each of the eight output data paths has associated with it an output channel record flag. This flag is a bit register with a separate set and a separate clear input. The flag is set when a record pulse is transmitted over the associated output cable. The flag is cleared when a resume pulse is returned over this output cable. This flag is also forced to a cleared position during a "dead start" condition.

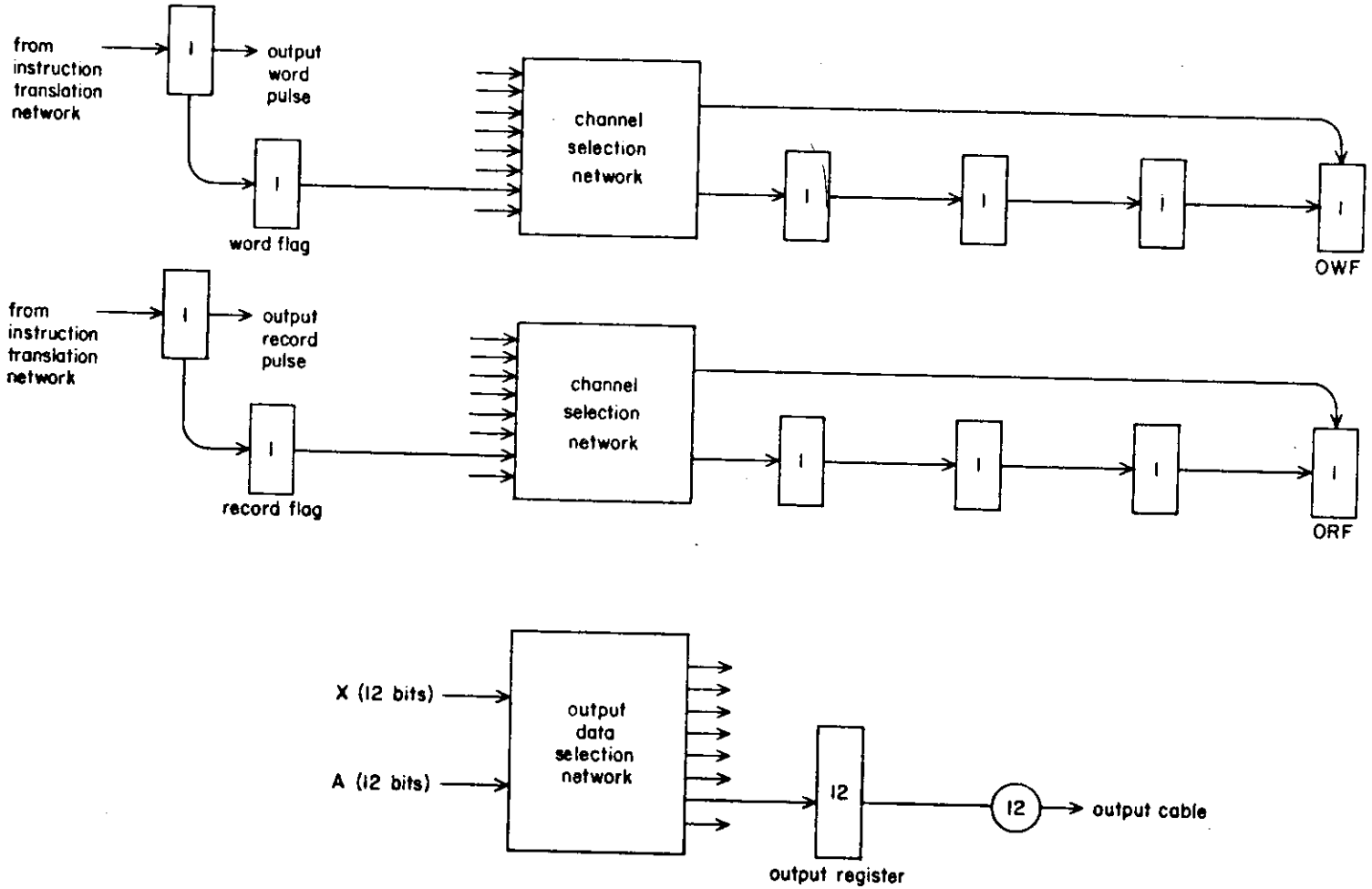


Fig. 4-9 Output Channels

A one clock period wide record pulse is formed for transmission over an output cable by a clear/enter type bit register which is cleared at the end of every clock period. This bit register is associated with the output channel record flag and sets the record flag in addition to transmitting the record pulse over the output cable. This bit register is set at the end of a clock period in which the following condition exists:

Condition: (f) = 74 & GRF & (d) = channel number

OWF synchronizing network

Figure 4-9 illustrates a selection and synchronizing network which connects the eight individual output channel word flags to a common output word flag (OWF). A channel selection network receives the data from the eight output channel word flags. This network selects one flag on the basis of the value in the lowest order three bits in the d register. The selected flag is then transmitted through a series of three synchronizing flags to the OWF. This synchronizing chain is necessary because the individual channel flags are cleared asynchronous with the PPU clock. The chain provides the time required to resolve partial pulses resulting at the timing interface. Each of the synchronizing flags is a clear/enter type bit register which is cleared at the end of every clock period. Each is set at the end of a clock period in which the preceding flag is set. The result is that the OWF is cleared four clock periods after the selected output channel word flag is cleared, or four clock periods after the d register selects a new channel, whichever is later.

ORF synchronizing network

Figure 4-9 illustrates a selection and synchronizing network which connects the eight individual output channel record flags to a common output record flag (ORF). This network is identical to the OWF synchronizing network and performs the same function for the output record flags that the OWF synchronizing network performs for the output word flags.

Output word flag (OWF)

The output word flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set by the OWF synchronizing network and presents a common control signal to the PPU for all output channel word flags. The OWF represents a synchronized copy of the currently selected output channel word flag. The OWF is set at the end of a clock period in which the selected channel output word flag is set, or the last flag in the OWF synchronizing network is set. The effect of this combination of conditions is that the OWF sets one clock period after the output channel word flag sets, and clears four clock periods after the output channel word flag clears. A new channel selection through the d register requires four clock periods for a correct interpretation of the newly selected output channel word flag.

Output record flag (ORF)

The output record flag is a clear/enter type bit register which is cleared at the end of every clock period. This flag is set by the ORF synchronizing network and presents a common control signal to the PPU for all output channel record flags. The ORF represents a synchronized copy of the currently selected output channel record flag. The ORF is set at the end of a clock period in which the selected channel output record flag is set, or the last flag in the ORF synchronizing network is set. The effect of this combination of conditions is that the ORF sets one clock period after the output channel record flag sets, and clears four clock periods after the output channel record flag clears. A new channel selection through the d register requires four clock periods for a correct interpretation of the newly selected output channel record flag.

Output data selection network

The output data selection network illustrated in figure 4-9 selects either the data in the X register, or the data in the lowest order 12 bits of the A register, for transmission to the eight output registers associated with the output channels. The data is sampled into one of the output registers when an output instruction is executed. The output data selection network is a static network. The A register data is selected when (f) = 70. The X register data is selected for all other conditions.

Channel output registers

There is a 12 bit clear/enter type register with gated clock pulse control associated with each output channel in the PPU. This register holds the output data from the time an output instruction is executed until a resume pulse is transmitted from the device receiving the data over the output cable. Each output register receives data from the output data selection network. An output register is cleared and new data is entered at the end of a clock period in which the following condition is present:

Condition: (f) = 72 & GRF & (d) = channel number
or: (f) = 73 & (k) = 2 & GRF & (d) = channel number

The data in the output register is transmitted statically over the output cable. This data remains on the cable until the register is cleared and new data is entered.

Full Duplex Communication

The PPUs communicate with the CPU and with other devices over channels which operate in a full duplex mode. That is, information may be transmitted from the PPU to a particular device at the same time that information is being received from that device. Each full duplex channel consists of an input data path and an output data path plus the associated control lines for each path. The full duplex channel consists of two physical cables. Each cable handles data moving in one direction and contains the control lines associated with that data. The two cables are completely symmetrical. Figure 4-10 illustrates the composition of the two cables in a full duplex channel.

Each cable contains 15 twisted pairs of wires plus ground wires. The data is transmitted over the twisted pairs in a differential mode. That is, the electrical current flowing on a given wire is compensated by an equal and opposite electrical current flowing on the other wire in the pair. There is a single transmitter for each wire pair and a single receiver. The receiver terminates the line in the characteristic impedance of the twisted pair. The transmitted electrical wave front is not reflected from the receiver back toward the transmitter. As a result, data may be transmitted at high frequency without regard to the length of the cable. The data is transmitted at a rate of 7.5 inches per nanosecond.

Word flag

A word flag is normally a 27.5 nanosecond pulse transmitted over a cable to notify the receiving device that the 12 data lines contain new information which is ready to be sampled. In special situations the word flag is forced to a continuously set condition. In this case the data may be sampled by the receiving device at any time. There is no coordination between transmitter and receiver in this case, and the receiver must interpret the data to extract information on time changes.

Record flag

A record flag is normally a 27.5 nanosecond pulse transmitted over a cable to notify the receiving device that a record of data transmission has been completed. In cases where the word flag is continuously set the record flag is not used.

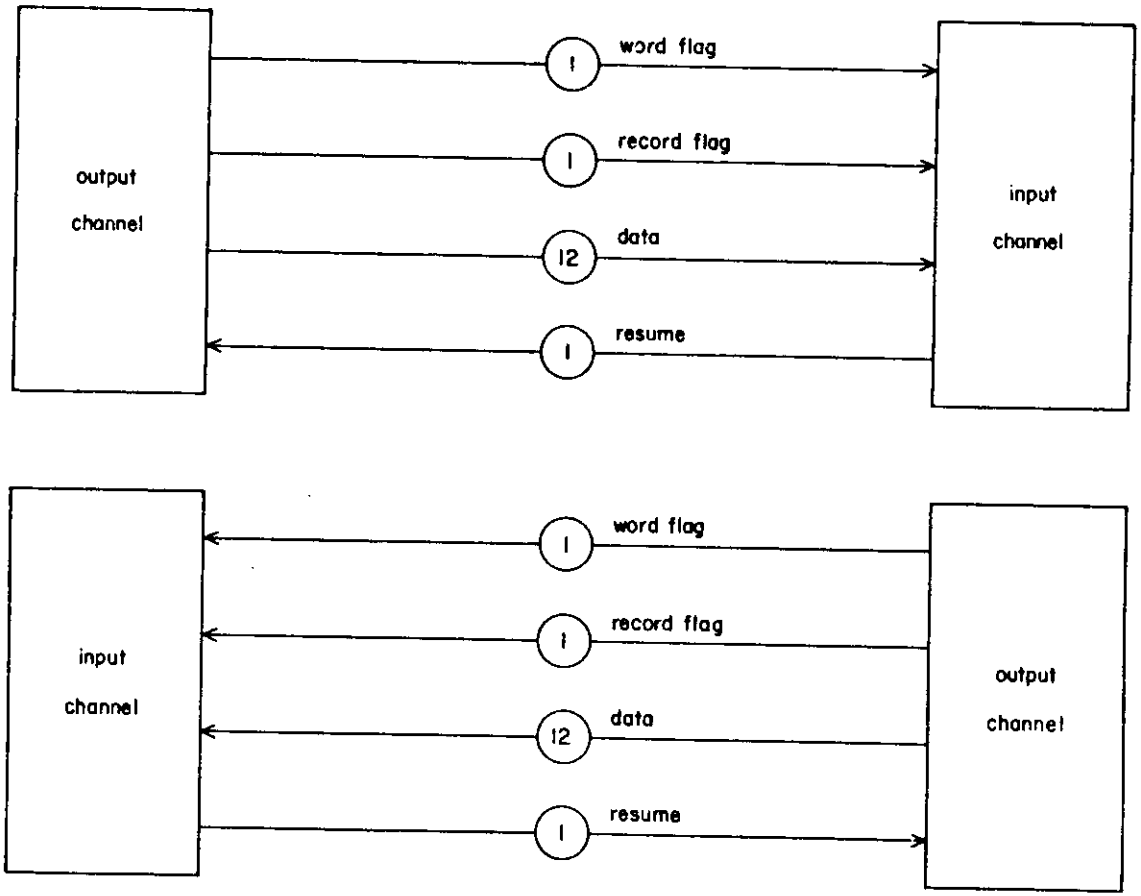


Fig. 4-10 Full Duplex Communication Channel

Resume

A resume is normally a 27.5 nanosecond pulse transmitted from the data receiving device back to the data transmitting device to indicate that the data on the cable has been sampled and the next data may be placed on the lines. In special situations the resume may be forced to a continuously set condition. In this case the data transmitting device may send new data at its own rate. There is no coordination between transmitter and receiver in this case, and the receiver must be ready to accept each 12 bit data word as transmitted.

Maximum cable length

The maximum cable length for 27.5 nanosecond pulses is 120 feet. For transmission over longer cables special circuits must be used to lengthen the pulse duration. An alternate mode forces the word flag and resume lines to continuous set conditions, and the data lines are used to include control information as well as data.

MCU Control Cable

The maintenance control unit (MCU) is connected to each PPU through a control cable. This cable is in addition to the eight sets of input and output data cables previously described. The control cable contains eight lines as illustrated in figure 4-11 on the following page. These are twisted pairs of wires similar to the lines used in the input and output data cables.

Dead start

One of the control lines between the PPU and the MCU is a "dead start" control line. The MCU program causes the transmission of a "dead start" signal over this line and results in a "dead start" condition in the PPU. The dead start condition lasts for several clock periods during which a number of the PPU register values are reset. At the end of the dead start interval the PPU is forced into a mode which loads a program over the PPU input channel zero. Data is entered in PPU storage beginning at address 0000. The data transmission over channel zero may be terminated by a record flag at any point in the loading of the PPU storage. If no record flag is set, the PPU will terminate the data input at 7777 octal words. In either case the PPU begins execution of the program at address 0001.

The following values are forced in the PPU registers by the dead start condition:

(A) = 007777
(P) = 0000
(X) = 0000
(f) = 71
(d) = 00
(k) = 1

In addition to the values forced in the registers, the PPU flags for the input and output channel control are all forced to a cleared condition. A continuous resume signal is sent from the PPU over all input cables during the dead start condition.

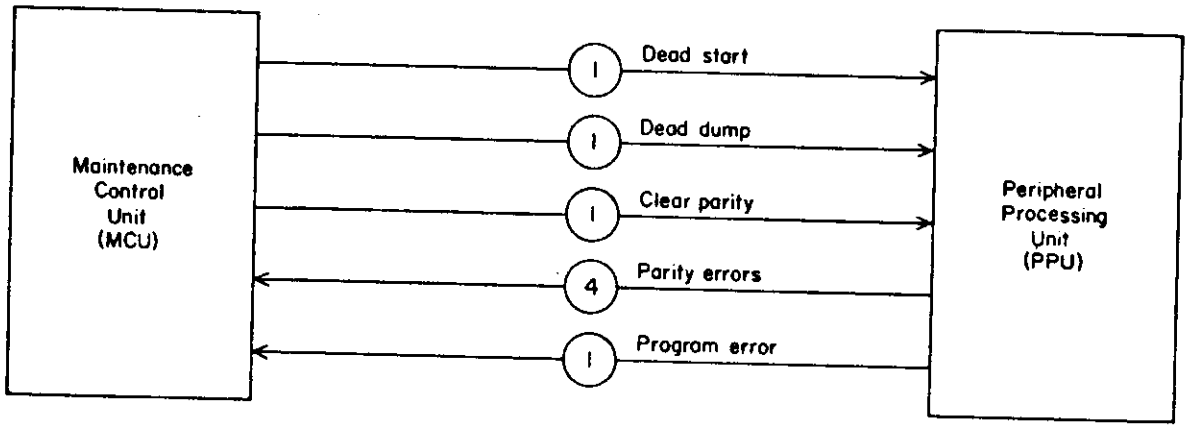


Fig. 4-11 MCU Control Cable

Dead dump

One of the control lines from the MCU to the PPU is a "dead dump" line. A signal is programed on this line when the PPU program has failed and a storage dump of PPU memory is desired to analyze the cause of failure. In this case a "dead start" signal must be programed first, followed by a "dead dump" signal. The dead dump signal is synchronized in the PPU and begins a data transmission of the entire PPU storage over output channel zero. Figure 4-12 illustrates the dead dump synchronizing chain, which consists of three clear/enter type bit registers which are cleared at the end of every clock period. The second bit in the f register is forced to a set condition at the end of every clock period in which the last flag in this timing chain is set. The effect of this condition is to change the 71 code from the "dead start" condition to a 73 code.

Parity error register

Each PPU contains a four bit parity error register as illustrated in figure 4-12. The register consists of bit registers with a separate set and a separate clear input. One of the four bits is set at the end of a clock period in which a parity error condition exists. A static selection network translates information from the storage sequence controls and S registers to decide which bit to set in order to indicate properly the storage stack in the PPU which had failed. A bit in the parity error register remains set until a "clear parity" condition is transmitted over the control cable from the MCU. This condition is programed in the MCU and clears all four bits in the selected PPU parity error register. The data in the parity error register is transmitted from the PPU to the MCU over the control cable.

Program error

A program error condition is transmitted from the PPU to the MCU over the control cable. This condition is a static translation of the f register contents. A signal is present on the following condition:

```
Condition:  (f) = 00
      or:   (f) = 77
```

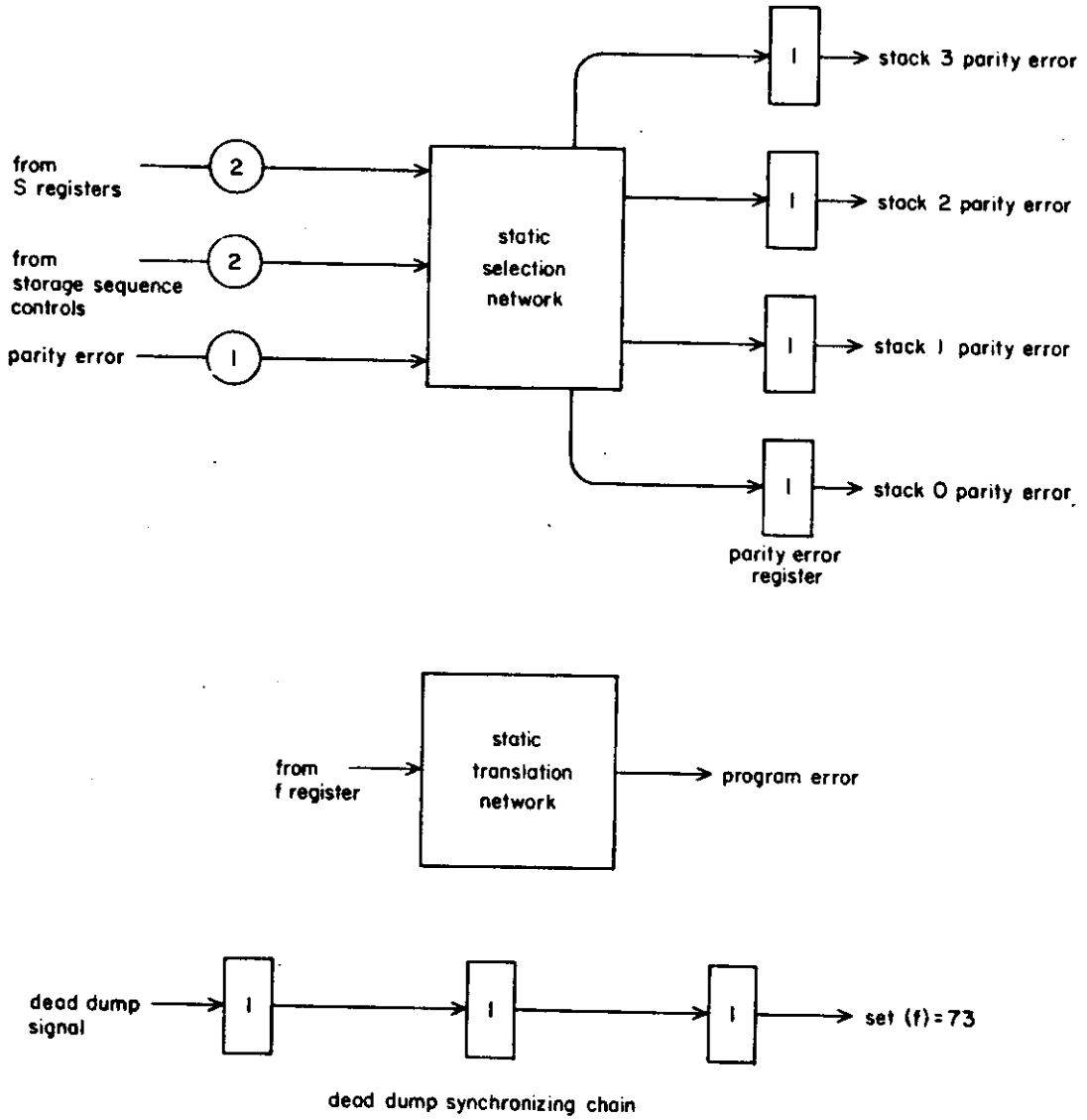


Fig. 4-12 Error Detection

5 PPU INSTRUCTIONS

PART 5 PPU INSTRUCTIONS

Introduction

This part of the reference manual describes the execution of the PPU instructions. Each instruction is described separately as to what it does, how long it takes to do it, and what happens if unusual or special situations arise. Most of the PPU instructions involve manipulation of internal registers in the PPU. The timing of execution for these instructions is dominated by the access time of the core storage banks. There are two independent banks of storage. One bank contains all of the even storage addresses and the other bank all of the odd storage addresses. If references to storage alternate between even and odd addresses each reference requires five clock periods. If two even references, or two odd references, occur consecutively the storage read/write cycle for the first reference must be completed before the second reference can begin. In this case a storage reference requires ten clock periods. As a result, the execution time for most of the PPU instructions is a multiple of five clock periods with variation in increments of five clock periods depending on the storage addresses involved.

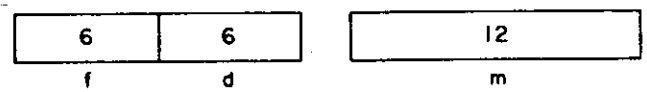
Input and output instructions require the presence of a control flag before the instruction execution can begin. Timing for the execution of these instructions is normally dominated by the response of devices external to the PPU. These instructions may be held in the f and d registers of the PPU for long periods of time waiting for a signal from an external device to begin execution of the instruction.

The shift instruction is the only PPU instruction in which the execution time is a function of the data involved. The shift is performed one bit position per clock period. As a result, the execution time for a shift instruction may vary from a storage dominated six clock periods to a shift count dominated 34 clock periods in one clock period increments.

The execution time for a PPU program loop can be precisely timed if no input or output instructions are involved and the storage addresses are known. A very close approximation can be obtained by ignoring the effect of the shift instruction and computing only the number and sequence of storage references.

Terminology

PPU instructions may occupy either one word or two words of storage space. In either case the highest order six bits of the first word contain the function code and are designated by the symbol f. The lowest order six bits in the first word designate an index register if one is used, or a channel number in the case of input or output instructions. This six bit quantity is always designated by the symbol d regardless of its use. The twelve bits in the second word of a two word format are designated by the symbol m. This is generally an address for a storage location. The format for a two word instruction is illustrated below.



PPU instruction format

The first 64 words in PPU storage are designated as index registers. These addresses are special in that they may be addressed by the d designator and their contents may be added to the m designator in an indexed storage reference. Care must be taken in the use of address 0000 for an index register. The value of d equal to zero is sensed as a special case in indexed references and is therefore not usable in this mode.

Special attention should be given to references to address 7777. All address arithmetic is performed in a ones complement mode. As a result address 0000 and 7777 refer to the same storage location. There are 4095 storage locations in a PPU memory, not 4096.

Shorthand notation in the instruction titles makes use of the symbol (d) to represent "the contents of index register d." The indirect instructions use the symbol ((d)). This represents "the contents of the storage location whose address is contained in index register d." This terminology is avoided in the instruction descriptions because of confusion with the d register in the hardware.

00XX Error Stop

This instruction format causes the PPU program execution to stop and to indicate a program error condition to the MCU. The PPU can be restarted only by a "dead start" condition.

0100 XXXX	Long jump to m

This instruction terminates the current program sequence with a jump to a new sequence beginning at address m. The value of d must be zero for this instruction. The instruction begins by reading the quantity m from storage location (P) + 1 to the X register. The address for the new program sequence is formed by adding (X) to a zero value in the Q register. This address is then used to fetch the first word of the new program sequence.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The second word of the current instruction must come from bank B. There can be no delay at CP00 due to a storage bank conflict. The storage reference at CP05 may be to either bank depending on the value of m. There will be a five clock period delay in the transmission of (Q) + (X) to S register if this reference is to bank B. The execution time for this instruction is 10 clock periods or 15 clock periods, depending on the value of m. Average execution time for this instruction is 12.5 clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 2

CP04 Read storage bank B to X register

CP05 Transmit (Q) + (X) to S register A

CP06 Begin read/write cycle for storage bank A

CP07 Transmit (P) + 1 to Q register
Transmit (Q) + (X) to P register
Set (k) to 0

CP09 Read storage bank A to X register
Read storage bank A to f and d registers

01XX XXXX	Long jump to m + (d)

This instruction terminates the current program sequence with a jump to a new sequence beginning at address m + (d). The value of d must be nonzero for this instruction. The instruction begins by reading the quantity m from storage location (P) + 1 and holding this quantity in the Q register. The content of index register d is then read into the X register. The address for the new program sequence is formed by adding (Q) to (X) in a 12 bit ones complement mode. The resulting address is used to fetch the first word of the new program sequence.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The second word of the current instruction must come from bank B. There can be no delay at CP00 due to a storage bank conflict. The storage reference at CP05 may be to either bank depending on the value of the d designator. There will be a five clock period delay in the transmission of (Q) to S register if this reference is to bank B. The storage reference at CP10 may be to either storage bank depending on the value of m + (d). There will be a five clock period delay in the transmission of (Q) + (X) to S register if this reference is to the same storage bank as CP05. There are three possible execution times for this instruction: 15 clock periods, 20 clock periods, or 25 clock periods, depending on the storage bank references. There is an equal probability of a 15 or 25 clock period execution time for random values. The average execution time for this instruction is therefore 20 clock periods.

Transmit (P) + 1 to S register B

Begin read/write cycle for storage bank B

Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 1

CP04 Read storage bank B to X register

CP05 Transmit (Q) to S register A

CP06 Begin read/write cycle for storage bank A

CP07 Transmit (X) to Q register
Set (k) to 2

Read storage bank A to X register

CP10 Transmit (Q) + (X) to S register B

Begin read/write cycle for storage bank B

CP12 Transmit (P) + 1 to Q register
Transmit (Q) + (X) to P register
Set (k) to 0

CP14 Read storage bank B to X register
Read storage bank B to f and d registers

0200 XXXX	Return jump to m

This instruction interrupts the current program sequence and inserts the execution of a subroutine between the current instruction in the present sequence and the following instruction. The called subroutine must have a common exit point in the form of a long jump to m instruction preceding the entry point. The return jump instruction inserts the exit address in the m location of the subroutine exit and then jumps to the entry point in the following word.

The value of d in this instruction must be zero. The instruction begins by reading the quantity m from storage location (P) + 1 to the Q register. This quantity is then used as a storage address to store (P) + 2 at storage location m. The first word of the new program sequence is then read from storage location m + 1.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The second word of the current instruction must come from bank B. There can be no delay at CP00 due to a storage bank conflict. The storage reference at CP05 may be to either bank depending on the value of m. There will be a five clock period delay in the transmission of (Q) + (X) to S register if this reference is to bank B. The storage reference at CP10 must be to a

different bank from the reference at CP05. As a result, there can be no delay at this point due to a storage bank conflict.

The execution time for this instruction is either 15 clock periods or 20 clock periods, depending on the value of m. Average execution time for this instruction is 17.5 clock periods.

```
CP00  Transmit (P) + 1 to S register B

CP01  Begin read/write cycle for storage bank B

      Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 2

      Read storage bank B to X register

      Transmit (Q) + (X) to S register A

CP06  Begin read/write cycle for storage bank A

      Transmit (P) + 1 to P register
      Transmit (Q) + (X) to Q register
      Set (k) to 3

CP09  Read storage bank A to X register

CP10  Transmit (P) to Z register A
      Transmit (Q) + 1 to S register B

CP11  Begin read/write cycle for storage bank B

CP12  Transmit (X) to Q register
      Transmit (Q) + 1 to P register
      Set (k) to 0

CP14  Read storage bank B to X register
      Read storage bank B to f and d registers
```

02XX XXXX Return jump to m + (d)

This instruction interrupts the current program sequence and inserts the execution of a subroutine between the current instruction in the present sequence and the following instruction. The called subroutine must have a common exit point in the form of a long jump to m instruction preceding the entry point. The return jump instruction inserts the exit address in the m location of the subroutine exit and then jumps to the entry point in the following word.

The value of d in this instruction must be nonzero. The instruction begins by reading the quantity m from storage location (P) + 1 to the Q register. The content of index register d is then read into the X register. The address for the new program sequence is formed by adding (Q) to (X) in a 12 bit ones complement mode. The resulting address is used to store (P) + 2 in the m field of the called subroutine exit instruction. The first word of the new program is then read from the following storage location.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The second word of the current instruction must come from bank B. There can be no delay at CP00 due to a storage bank conflict. The storage reference at CP05 may be to either bank depending on the value of d. There will be a five clock period delay in the transmission of (Q) to S register if this reference is to bank B. The storage reference at CP10 may be to either bank depending on the value of m + (d). There will be a five clock period delay in the transmission of (Q) + (X) to S register if this reference is to the same bank as referenced in the CP05 case. The storage reference at CP15 must be to a different bank from that referenced at CP10. There can be no delay at this point due to a storage conflict.

The execution time for this instruction may be 20 clock periods, 25 clock periods, or 30 clock periods, depending on the values of d and m + (d). Average execution time for this instruction is 25 clock periods.

```
CP00  Transmit (P) + 1 to S register B

CP01  Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 1

CP04  Read storage bank B to X register

CP05  Transmit (Q) to S register A

CP06  Begin read/write cycle for storage bank A

CP07  Transmit (X) to Q register
      Set (k) to 2

CP09  Read storage bank A to X register

CP10  Transmit (Q) + (X) to S register B

CP11  Begin read/write cycle for storage bank B

CP12  Transmit (P) + 1 to P register
      Transmit (Q) + (X) to Q register
      Set (k) to 3

CP14  Read storage bank B to X register

CP15  Transmit (P) to Z register B
      Transmit (Q) + 1 to S register A

CP16  Begin read/write cycle for storage bank A

CP17  Transmit (X) to Q register
      Transmit (Q) + 1 to P register
      Set (k) to 0

CP19  Read storage bank A to X register
      Read storage bank A to f and d registers
```

03XX Unconditional jump d

This instruction interrupts the current program sequence with a jump to a new sequence beginning at an address incrementally related to the current program address. The d designator may specify a new sequence which begins at an address either forward or backward from the current address by an amount not greater than 31 decimal locations. The d designator is considered as a six bit ones complement number in determining the increment for the jump.

As an example consider a d value of 16 octal. The new program sequence in this case will begin with an instruction word located 16 octal locations beyond the location of the 03XX instruction. Now consider a d value of 55 octal. The new program sequence in this case will begin with an instruction word located 22 octal locations before the location of the 03XX instruction. Values of 00 and 77 for the d designator must not be used with this instruction. These two values cause the peripheral processor program to lock up and require dead starting the system with a new program.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. A three clock period delay occurs at the beginning of this sequence to form the jump address. The storage reference at CP03 may be to either bank depending on the value of d. There will be a two clock period delay in the transmission of (P) + (d) to S register if this reference is to bank A. The execution time for this instruction is 8 clock periods or 10 clock periods, depending on the value of d. Average execution time for this instruction is 9 clock periods.

CP03 Transmit (P) + (d) to S register B

CP04 Begin read/write cycle for storage bank B

CP05 Transmit (d) to Q register
Transmit (P) + (d) to P register
Set (k) to 0

CP07 Read storage bank B to X register
Read storage bank B to f and d registers

04XX Zero jump d

This is a conditional branch instruction which continues the current program sequence or jumps to a new program sequence, depending on the content of the A register. If (A) = 000000 the current program sequence is terminated with a jump to an address specified by the d designator. If (A) ≠ 000000 the current program sequence continues with the execution of the next instruction. A value of (A) = 777777 is not considered as zero for this instruction.

If the jump is taken, the new program sequence begins at an address either forward or backward from the current address by an amount not greater than 31 decimal locations. The d designator is considered as a six bit ones complement number in determining the increment for the jump. (See instruction 03XX for examples.)

Execution time

There are two possible timing sequences for this instruction. The first sequence is listed below and represents the timing for the case where the jump is not taken. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 must be to bank B in this case, and no delay can occur as a result of bank conflict. Execution time for this case is always five clock periods.

```
      Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

      Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

CP04  Read storage bank B to X register
      Read storage bank B to f and d registers
```

The second sequence represents the timing for the case where the jump is taken. This sequence is identical to that listed for the 03XX instruction.

05XX Nonzero jump d

This is a conditional branch instruction which continues the current program sequence or jumps to a new program sequence, depending on the content of the A register. If (A) ≠ 000000 the current program sequence is terminated with a jump to an address specified by the d designator. If (A) = 000000 the current program sequence continues with the execution of the next instruction. A value of (A) = 777777 is not considered as zero for this instruction.

If the jump is taken, the new program sequence begins at an address either forward or backward from the current address by an amount not greater than 31 decimal locations. The d designator is considered as a six bit ones complement number in determining the increment for the jump. (See instruction 03XX for examples.)

Execution time

There are two possible timing sequences for this instruction. The first sequence is listed below and represents the timing for the case where the jump is not taken. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 must be to bank B in this case, and no delay can occur as a result of bank conflict. Execution time for this case is always five clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 0

CP04 Read storage bank B to X register
Read storage bank B to f and d registers

The second sequence represents the timing for the case where the jump is taken. This sequence is identical to that listed for the 03XX instruction.

06XX Positive jump d

This is a conditional branch instruction which continues the current program sequence or jumps to a new program sequence, depending on the content of the A register. If the highest order bit in the A register has a zero value, the current program sequence is terminated with a jump to an address specified by the d designator. If the highest order bit in the A register has a one value, the current program sequence continues with the execution of the next instruction.

If the jump is taken, the new program sequence begins at an address either forward or backward from the current address by an amount not greater than 31 decimal locations. The d designator is considered as a six bit ones complement number in determining the increment for the jump. (See instruction 03XX for examples.)

Execution time

There are two possible timing sequences for this instruction. The first sequence is listed below and represents the timing for the case where the jump is not taken. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 must be to bank B in this case, and no delay can occur as a result of bank conflict. Execution time for this case is always five clock periods.

```
Transmit (P) + 1 to S register B

Begin read/write cycle for storage bank B

Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 0

Read storage bank B to X register
Read storage bank B to f and d registers
```

The second sequence represents the timing for the case where the jump is taken. This sequence is identical to that listed for the 03XX instruction.

07XX Negative jump d

This is a conditional branch instruction which continues the current program sequence or jumps to a new program sequence, depending on the content of the A register. If the highest order bit in the A register has a one value, the current program sequence is terminated with a jump to an address specified by the d designator. If the highest order bit in the A register has a zero value, the current program sequence continues with the execution of the next instruction.

If the jump is taken, the new program sequence begins at an address either forward or backward from the current address by an amount not greater than 31 decimal locations. The d designator is considered as a six bit ones complement number in determining the increment for the jump. (See instruction 03XX for examples.)

Execution time

There are two possible timing sequences for this instruction. The first sequence is listed below and represents the timing for the case where the jump is not taken. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 must be to bank B in this case, and no delay can occur as a result of bank conflict. Execution time for this case is always five clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 0

CP04 Read storage bank B to X register
Read storage bank B to f and d registers

The second sequence represents the timing for the case where the jump is taken. This sequence is identical to that listed for the 03XX instruction.

10XX Shift d

This instruction shifts the contents of the A register either to the right open ended or to the left circularly as specified by the d designator. The d designator is treated as a six bit ones complement number in this instruction. If the highest order bit in the d designator is zero the content of the A register is shifted circularly to the left by the number of bit positions indicated in the value of the d designator. If the highest order bit in the d designator is one the content of the A register is shifted open ended to the right by the complement of the value of the d designator.

In a left circular shift the content of the A register is shifted one bit position at a time. In each shift the lowest order bit position in the register is filled by the bit previously held in the highest order bit position. No bits are lost in this process but are repositioned toward the higher order bit positions. A d designator value of 00 causes no shift to take place. A d designator value greater than 18 decimal causes the content of the A register to shift completely around the A register. A maximum of a 31 decimal shift count may be used.

In a right open ended shift the content of the A register is shifted one bit position at a time toward the lower order bit positions in the register. The highest order bit position in the A register is filled with a zero value as each shift occurs. The lowest order bit in the A register is discarded as each shift occurs. A maximum of a 31 decimal shift count may be used. For all shift counts larger than 17 decimal the final A register value will be 000000. A designator value of 77 will cause no shift to take place.

Execution time

The execution time for this instruction depends on the number of shifts performed in the instruction. Each shift requires one clock period. The fetch of the next instruction begins when the shift count has been reduced to a value of three or less. Shift counts of three or less, therefore, result in a minimum execution time. The execution time for the instruction increases by one clock period for each additional shift required beyond three. Minimum execution time is six clock periods. Maximum execution time is 34 clock periods.

A timing sequence is listed below for a 1005 instruction. This instruction will shift the A register content left circularly by five bit positions. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The next storage reference must be to bank B, and no delay can occur due to storage bank conflict regardless of shift count. A one clock period delay always occurs in execution of the 10XX instruction to transmit the shift count to the sk register. This causes the six clock period minimum execution time for the instruction. For the case listed below an additional two clock period delay results from the shift count larger than three.

```
CP00  Transmit (d) to sk register

CP01  Shift (A) left circularly by one bit position
      Reduce (sk) by one count (from 5 to 4)

CP02  Shift (A) left circularly by one bit position
      Reduce (sk) by one count (from 4 to 3)

CP03  Transmit (P) + 1 to S register B
      Shift (A) left circularly by one bit position
      Reduce (sk) by one count (from 3 to 2)

CP04  Begin read/write cycle for storage bank B
      Shift (A) left circularly by one bit position
      Reduce (sk) by one count (from 2 to 1)

CP05  Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Shift (A) left circularly by one bit position
      Reduce (sk) by one count (from 1 to 0)
      Set (k) to 0

CP07  Read storage bank B to X register
      Read storage bank B to f and d registers
```

11XX Logical difference d

This instruction forms in the A register the logical difference of the original (A) and the d designator considered as a six bit positive integer. The highest order 12 bits in the A register are not affected by this operation.

The logical difference is formed on a bit by bit basis. The logical difference of two binary quantities A and B is defined to be:

A .and..not. B .or. B .and..not. A

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The next instruction must come from bank B. There can be no delay at CP00 because of a storage bank conflict. Execution time for this instruction is five clock periods.

```
CP00  Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

      Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Transmit (A) .ld. (d) to A register
      Set (k) to 0

      Read storage bank B to X register
      Read storage bank B to f and d registers
```

12XX Logical product d

This instruction forms in the A register the logical product of the original (A) and the d designator considered as a six bit positive integer. The highest order 12 bits in the A register are always cleared to zero by this instruction.

The logical product is formed on a bit by bit basis. The logical product of two binary quantities A and B is defined to be:

A .and. B

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The next instruction must come from bank B. There can be no delay at CP00 because of a storage bank conflict. Execution time for this instruction is five clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Transmit (A) .lp. (d) to A register
Set (k) to 0

CP04 Read storage bank B to X register
Read storage bank B to f and d registers

13XX Selective clear d

This instruction forms in the A register the logical product of the original (A) and the complement of the d designator considered as a six bit positive integer. The highest order 12 bits in the A register are not affected by this instruction.

This operation is performed on a bit by bit basis. The function performed on a pair of binary quantities A and B is defined to be:

A .and..not. B

where A corresponds to the A register bit.

Execution time

The timing sequence for this instruction i listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The next instruction must come from bank B. There can be no delay at CP00 because of a storage bank conflict. Execution time for this instruction is five clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Transmit (A) .sc. (d) to A register
Set (k) to 0

CP04 Read storage bank B to X register
Read storage bank B to f and d registers

14XX Load d

This instruction enters in the A register a copy of the d designator considered as a six bit positive integer. The highest order 12 bits in the A register are always cleared to zero by this instruction.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The next instruction must come from bank B. There can be no delay at CP00 because of a storage bank conflict. Execution time for this instruction is five clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Transmit (d) to A register
Set (k) to 0

CP04 Read storage bank B to X register
Read storage bank B to f and d registers

15XX Load complement d

This instruction enters in the A register a complemented copy of the d designator. The highest order 12 bits in the A register are always set to one by this instruction. The lowest order six bits are bit by bit complements of the corresponding bits in the d designator.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The next instruction must come from bank B. There can be no delay at CP00 because of a storage bank conflict. Execution time for this instruction is five clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Transmit -(d) to A register
Set (k) to 0

CP04 Read storage bank B to X register
Read storage bank B to f and d registers

16XX Add d

This instruction adds the d designator, considered as a six bit positive quantity, to the current contents of the A register. The result is left in the A register. The addition is performed in an 18 bit ones complement mode. An 18 bit operand is formed from the d designator by adding 12 higher order zero bits.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The next instruction must come from bank B. There can be no delay at CP00 because of a storage bank conflict. Execution time for this instruction is five clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Transmit (A) + (d) to A register
Set (k) to 0

CP04 Read storage bank B to X register
Read storage bank B to f and d registers

17XX Subtract d

This instruction subtracts the d designator, considered as a six bit positive quantity, from the current contents of the A register. The result is left in the A register. An 18 bit operand is formed from the d designator. This operand consists of 12 one bits in the highest order bit positions and six lowest order bits which are bit by bit complements of the corresponding bits in the d designator. This 18 bit operand is added to the original contents of the A register in an 18 bit ones complement mode.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The next instruction must come from bank B. There can be no delay at CP00 because of a storage bank conflict. Execution time for this instruction is five clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Transmit (A) - (d) to A register
Set (k) to 0

CP04 Read storage bank B to X register
Read storage bank B to f and d registers

20XX XXXX Load dm

This instruction clears the A register and enters an 18 bit operand consisting of the d and m designators. The d designator is inserted in the highest order six bit positions of the A register. The m designator is inserted in the lowest order 12 bit positions in the A register.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for the second word in the current instruction. The storage reference at CP05 must be to bank A for the first word of the next instruction. There can be no delay due to storage bank conflict at either of these storage references. Execution time for this instruction is always ten clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 1

CP04 Read storage bank B to X register

CP05 Transmit (P) + 1 to S register A

CP06 Begin read/write cycle for storage bank A

CP07 Transmit (QX) to A register
Transmit (P) + 1 to P register
Transmit (X) to Q register
Set (k) to 0

CP09 Read storage bank A to X register
Read storage bank A to f and d registers

21XX XXXX Add dm

This instruction adds an 18 bit operand consisting of the d and m designators to the current contents of the A register. The result is left in the A register. The addition is performed in an 18 bit ones complement mode. The d designator forms the highest order six bits of the 18 bit operand and the m designator completes the lowest order 12 bits.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for the second word in the current instruction. The storage reference at CP05 must be to bank A for the first word of the next instruction. There can be no delay due to storage bank conflict at either of these storage references. Execution time for this instruction is always ten clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 1

CP04 Read storage bank B to X register

CP05 Transmit (P) + 1 to S register A

CP06 Begin read/write cycle for storage bank A

CP07 Transmit (A) + (QX) to A register
Transmit (P) + 1 to P register
Transmit (X) to Q register
Set (k) to 0

CP09 Read storage bank A to X register
Read storage bank A to f and d registers

22XX XXXX Logical product dm

This instruction forms the bit by bit logical product of (A) and an 18 bit operand consisting of the d and m designators. The result is left in the A register. The d designator forms the highest order six bits of the 18 bit operand and the m designator completes the lowest order 12 bits.

The logical product of two binary quantities A and B is defined to be:

A .and. B

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for the second word in the current instruction. The storage reference at CP05 must be to bank A for the first word of the next instruction. There can be no delay due to storage bank conflict at either of these storage references. Execution time for this instruction is always ten clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 1

CP04 Read storage bank B to X register

CP05 Transmit (P) + 1 to S register A

CP06 Begin read/write cycle for storage bank A

CP07 Transmit (A) .lp. (QX) to A register
Transmit (P) + 1 to P register
Transmit (X) to Q register
Set (k) to 0

CP09 Read storage bank A to X register
Read storage bank A to f and d registers

23XX XXXX Logical difference dm

This instruction forms the bit by bit logical difference of (A) and an 18 bit operand consisting of the d and m designators. The result is left in the A register. The d designator forms the highest order six bits of the 18 bit operand and the m designator completes the lowest order 12 bits.

The logical difference of two binary quantities A and B is defined to be:

A .and..not. B .or. B .and..not. A

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for the second word in the current instruction. The storage reference at CP05 must be to bank A for the first word of the next instruction. There can be no delay due to storage bank conflict at either of these storage references. Execution time for this instruction is always ten clock periods.

```
      Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 1

CP04  Read storage bank B to X register

CP05  Transmit (P) + 1 to S register A

      Begin read/write cycle for storage bank A

      Transmit (A) .ld. (QX) to A register
      Transmit (P) + 1 to P register
      Transmit (X) to Q register
      Set (k) to 0

CP09  Read storage bank A to X register
      Read storage bank A to f and d registers
```

24XX	Pass
25XX	Pass
26XX	Pass
27XX	Pass

These four instructions are identical and perform no logical function. Each instruction results in a five clock period delay.

Execution time

The timing sequence for these instructions is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 must be to bank B for the next instruction word. There can be no delay due to storage bank conflict at this storage reference.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 0

CP04 Read storage bank B to X register
Read storage bank B to f and d registers

30XX Load (d)

This instruction clears the A register and enters a 12 bit operand from index register d. The operand is entered in the A register as a 12 bit positive integer. The highest order six bits in the A register are always cleared by this instruction.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. If this reference is to bank A a five clock period delay will occur in the transmission of (d) to S register. However, in this case no delay will occur at CP05 for the storage reference for the next instruction in bank B. If the storage reference at CP00 is to bank B no delay will occur in the transmission of (d) to S register. A five clock period delay will then occur at CP05 in the transmission of (P) + 1 to S register for the next instruction word. This second situation is illustrated in the timing sequence below. As a result of these two possible sequences the execution time for this instruction is a constant 15 clock periods.

```
      Transmit (d) to S register B

      Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Set (k) to 1

      Read storage bank B to X register

CP10  Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

      Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Transmit (X) to A register
      Set (k) to 0

      Read storage bank B to X register
      Read storage bank B to f and d registers
```

31XX Add (d)

This instruction adds the content of index register d considered as a 12 bit positive quantity to the current contents of the A register. The result is left in the A register. The addition is performed in an 18 bit ones complement mode. An 18 bit operand is formed from the index register contents by adding six higher order zero bits.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. If this reference is to bank A a five clock period delay will occur in the transmission of (d) to S register. However, in this case no delay will occur at CP05 for the storage reference for the next instruction in bank B. If the storage reference at CP00 is to bank B no delay will occur in the transmission of (d) to S register. A five clock period delay will then occur at CP05 in the transmission of (P) + 1 to S register for the next instruction word. This second situation is illustrated in the timing sequence below. As a result of these two possible sequences the execution time for this instruction is a constant 15 clock periods.

```
      Transmit (d) to S register B

      Begin read/write cycle for storage bank B

      Transmit (d) to Q register
      Set (k) to 1

CP04  Read storage bank B to X register

      Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

CP12  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Transmit (A) + (X) to A register
      Set (k) to 0

      Read storage bank B to X register
      Read storage bank B to f and d registers
```

32XX Subtract (d)

This instruction subtracts the content of index register d considered as a 12 bit positive quantity from the current contents of the A register. The result is left in the A register. The operation is performed by adding the complement of the index register contents to (A) in an 18 bit ones complement mode. An 18 bit operand is formed for the addition by forcing the highest order six bits to a one value The lowest order 12 bits are the bit by bit complement of the index register contents.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. If this reference is to bank A a five clock period delay will occur in the transmission of (d) to S register. However, in this case no delay will occur at CP05 for the storage reference for the next instruction in bank B. If the storage reference at CP00 is to bank B no delay will occur in the transmission of (d) to S register. A five clock period delay will then occur at CP05 in the transmission of (P) + 1 to S register for the next instruction word. This second situation is illustrated in the timing sequence below. As a result of these two possible sequences the execution time for this instruction is a constant 15 clock periods.

```
      Transmit (d) to S register B

      Begin read/write cycle for storage bank B

      Transmit (d) to Q register
      Set (k) to 1

CP04  Read storage bank B to X register

CP10  Transmit (P) + 1 to S register B

CP11  Begin read/write cycle for storage bank B

CP12  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Transmit (A) - (X) to A register
      Set (k) to 0

CP14  Read storage bank B to X register
      Read storage bank B to f and d registers
```

33XX Logical difference (d)

This instruction forms in the A register the logical difference of the content of index register d considered as a 12 bit positive quantity and the original (A). The highest order six bits in the A register are not affected by this operation.

The logical difference is formed on a bit by bit basis. The logical difference of two binary quantities A and B is defined to be:

A .and..not. B .or. B .and..not. A

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. If this reference is to bank A a five clock period delay will occur in the transmission of (d) to S register. However, in this case no delay will occur at CP05 for the storage reference for the next instruction in bank B. If the storage reference at CP00 is to bank B no delay will occur in the transmission of (d) to S register. A five clock period delay will then occur at CP05 in the transmission of (P) + 1 to S register for the next instruction word. This second situation is illustrated in the timing sequence below. As a result of these two possible sequences the execution time for this instruction is a constant 15 clock periods.

CP00 Transmit (d) to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Set (k) to 1

Read storage bank B to X register

Transmit (P) + 1 to S register B

CP11 Begin read/write cycle for storage bank B

CP12 Transmit (X) to Q register
Transmit (P) + 1 to P register
Transmit (A) .ld. (X) to A register
Set (k) to 0

Read storage bank B to X register
Read storage bank B to f and d registers

34XX Store (d)

This instruction stores the lowest order 12 bits of (A) in index register d. The content of the A register is not altered in this process.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. If this reference is to bank A a five clock period delay will occur in the transmission of (d) to S register. However, in this case no delay will occur at the storage reference for the next instruction in bank B. If the storage reference at CP00 is to bank B no delay will occur in the transmission of (d) to S register. A five clock period delay will then occur in the transmission of (P) + 1 to S register for the next instruction word. This second situation is illustrated in the timing sequence below. As a result of the delays in the two possible sequences the execution time for this instruction is a constant 15 clock periods.

CP00 Transmit (d) to S register B

Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Set (k) to 1

Read storage bank B to X register

Transmit (A) to Z register B

Transmit (P) + 1 to S register B

Begin read/write cycle for storage bank B

Transmit (X) to Q register
Transmit (P) + 1 to P register
Set (k) to 0

CP14 Read storage bank B to X register
Read storage bank B to f and d registers

35XX Replace add (d)

This instruction adds the content of index register d considered as a 12 bit positive quantity to the current contents of the A register. The result is left in the A register and is also stored in index register d. The addition is performed in an 18 bit ones complement mode. An 18 bit operand is formed from the index register contents by adding six higher order zero bits. The result stored in index register d is the lowest order 12 bits of the resulting 18 bit sum.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. If this reference is to bank A a five clock period delay will occur in the transmission of (d) to S register. However, in this case no delay will occur at the storage reference for the next instruction in bank B. If the storage reference at CP00 is to bank B no delay will occur in the transmission of (d) to S register. A five clock period delay will then occur in the transmission of (P) + 1 to S register for the next instruction word. This second situation is illustrated in the timing sequence below. In either possible sequence there will be a five clock period delay between the reading of (d) and the storing of the result in the same storage location. As a result of these conditions the execution time for the instruction is a constant 25 clock periods.

CP00 Transmit (d) to S register B

CP01 Begin read/write cycle for storage bank B

Transmit (d) to Q register
Set (k) to 1

CP04 Read storage bank B to X register

CP10 Transmit (Q) to S register B

CP11 Begin read/write cycle for storage bank B

Transmit (A) + (X) to A register
Set (k) to 2

CP14 Read storage bank B to X register

Transmit (A) to Z register B

Transmit (P) + 1 to S register B

CP21 Begin read/write cycle for storage bank B

CP22 Transmit (X) to Q register
Transmit (P) + 1 to P register
Set (k) to 0

Read storage bank B to X register
Read storage bank B to f and d registers

36XX Replace add one (d)

This instruction increases the contents of index register d by one count. Execution begins by clearing the A register and entering a value of plus one. The contents of index register d are read from storage to the X register and then added to (A) in an 18 bit ones complement mode. The index register value is treated as a 12 bit positive quantity in this process. An 18 bit operand is formed from (X) by adding six higher order zero bits. The result is left in the A register, and the lowest order 12 bits are stored in index register d. Note that the arithmetic is essentially twos complement as viewed by the index register, and the quantity in the A register is not necessarily equal to the result in the index register.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. If this reference is to bank A a five clock period delay will occur in the transmission of (d) to S register. However, in this case no delay will occur at the storage reference for the next instruction in bank B. If the storage reference at CP00 is to bank B no delay will occur in the transmission of (d) to S register. A five clock period delay will then occur in the transmission of (P) + 1 to S register for the next instruction word. This second situation is illustrated in the timing sequence below. In either possible sequence there will be a five clock period delay between the reading of (d) and the storing of the result in the same storage location. As a result of these conditions the execution time for the instruction is a constant 25 clock periods.

```
CP00  Transmit (d) to S register B

CP01  Begin read/write cycle for storage bank B

      Transmit +1 to A register
      Transmit (d) to Q register
      Set (k) to 1

      Read storage bank B to X register

CP10  Transmit (Q) to S register B

      Begin read/write cycle for storage bank B

CP12  Transmit (A) + (X) to A register
      Set (k) to 2

CP14  Read storage bank B to X register

CP15  Transmit (A) to Z register B

CP20  Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

      Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

CP24  Read storage bank B to X register
      Read storage bank B to f and d registers
```

37XX Replace subtract one (d)

This instruction decreases the contents of index register d by one count. Execution begins by clearing the A register and entering a value of minus one. The contents of index register d are read from storage to the X register and then added to (A) in an 18 bit ones complement mode. The index register value is treated as a 12 bit positive quantity in this process. An 18 bit operand is formed from (X) by adding six higher order zero bits. The result is left in the A register, and the lowest order 12 bits are stored in index register d. Note that the arithmetic is essentially twos complement as viewed by the index register, and the quantity in the A register is not necessarily equal to the result in the index register.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. If this reference is to bank A a five clock period delay will occur in the transmission of (d) to S register. However, in this case no delay will occur at the storage reference for the next instruction in bank B. If the storage reference at CP00 is to bank B no delay will occur in the transmission of (d) to S register. A five clock period delay will then occur in the transmission of (P) + 1 to S register for the next instruction word. This second situation is illustrated in the timing sequence below. In either possible sequence there will be a five clock period delay between the reading of (d) and the storing of the result in the same storage location. As a result of these conditions the execution time for the instruction is a constant 25 clock periods.

```
CP00  Transmit (d) to S register B

      Begin read/write cycle for storage bank B

CP02  Transmit -1 to A register
      Transmit (d) to Q register
      Set (k) to 1

      Read storage bank B to X register

CP10  Transmit (Q) to S register B

      Begin read/write cycle for storage bank B

CP12  Transmit (A) + (X) to A register
      Set (k) to 2

      Read storage bank B to X register

      Transmit (A) to Z register B

      Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

      Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

      Read storage bank B to X register
      Read storage bank B to f and d registers
```

40XX Load ((d))

This instruction clears the A register and enters a 12 bit operand from storage. The address for the operand is contained in index register d. The highest order six bits in the A register are always cleared by this instruction.

Instruction execution begins with a storage reference to index register d. The contents of this index register are read into the X register. A second storage reference is then made using (X) as the storage address. This operand is read into the A register, and the highest order six bits in the A register are cleared. A third storage reference is then initiated to read the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. The storage reference at CP05 may be to either bank depending on the contents of index register d. These two references may occur with all four combinations of possible bank sequences. The timing sequence illustrated below is the shortest of the four possible combinations. Execution time for this case is 15 clock periods. The other three possible combinations result in delays at different points in the sequence, but in each case there are two delays total in the execution of the instruction. The result is that these three cases have an execution time of 25 clock periods. Average execution time for the instruction, assuming equal probability of the four cases, is 22.5 clock periods.

Transmit (d) to S register B

Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Set (k) to 1

CP04 Read storage bank B to X register

Transmit (X) to S register A

CP06 Begin read/write cycle for storage bank A

CP07 Transmit (X) to Q register
Set (k) to 2

Read storage bank A to X register

Transmit (P) + 1 to S register B

Begin read/write cycle for storage bank B

Transmit (X) to Q register
Transmit (P) + 1 to P register
Transmit (X) to A register
Set (k) to 0

Read storage bank B to X register
Read storage bank B to f and d registers

41XX Add ((d))

This instruction reads an operand from storage and adds it to the current contents of the A register. The addition is performed in an 18 bit ones complement mode. An 18 bit operand is formed from the 12 bit storage operand by adding six higher order zero bits. The address for the operand is contained in index register d.

Instruction execution begins with a storage reference to index register d. The contents of this index register are read into the X register. A second storage reference is then made using (X) as the storage address. This operand is read into the X register and then added to the contents of the A register. A third storage reference is then initiated to read the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. The storage reference at CP05 may be to either bank depending on the contents of index register d. These two references may occur with all four combinations of possible bank sequences. The timing sequence illustrated below is the shortest of the four possible combinations. Execution time for this case is 15 clock periods. The other three possible combinations result in delays at different points in the sequence, but in each case there are two delays total in the execution of the instruction. The result is that these three cases have an execution time of 25 clock periods. Average execution time for the instruction, assuming equal probability of the four cases, is 22.5 clock periods.

```
CP00  Transmit (d) to S register B

CP01  Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Set (k) to 1

CP04  Read storage bank B to X register

      Transmit (X) to S register A

CP06  Begin read/write cycle for storage bank A

CP07  Transmit (X) to Q register
      Set (k) to 2

CP09  Read storage bank A to X register

CP10  Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

CP12  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Transmit (A) + (X) to A register
      Set (k) to 0

      Read storage bank B to X register
      Read storage bank B to f and d registers
```

42XX	Subtract ((d))

This instruction reads an operand from storage and subtracts it from the current contents of the A register. The result is left in the A register. The address for the operand is contained in index register d. The operation is performed by adding the complement of the operand to (A) in an 18 bit ones complement mode. An 18 bit operand for the addition is formed from the 12 bit storage operand by forcing the highest order six bits to a one value. The lowest order 12 bits are the bit by bit complement of the storage operand values.

Instruction execution begins with a storage reference to index register d. The contents of this index register are read into the X register. A second storage reference is then made using (X) as the storage address This operand is read into the X register and then subtracted from the contents of the A register. A third storage reference is then initiated to read the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. The storage reference at CP05 may be to either bank depending on the contents of index register d. These two references may occur with all four combinations of possible bank sequences. The timing sequence illustrated below is the shortest of the four possible combinations. Execution time for this case is 15 clock periods. The other three possible combinations result in delays at different points in the sequence, but in each case there are two delays total in the execution of the instruction. The result is that these three cases have an execution time of 25 clock periods. Average execution time for the instruction, assuming equal probability of the four cases, is 22.5 clock periods.

```
      Transmit (d) to S register B

      Begin read/write cycle for storage bank B

      Transmit (d) to Q register
      Set (k) to 1

CP04  Read storage bank B to X register

CP05  Transmit (X) to S register A

CP06  Begin read/write cycle for storage bank A

      Transmit (X) to Q register
      Set (k) to 2

CP09  Read storage bank A to X register

CP10  Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

      Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Transmit (A) - (X) to A register
      Set (k) to 0

CP14  Read storage bank B to X register
      Read storage bank B to f and d registers
```

43XX Logical difference ((d))

This instruction forms in the A register the logical difference of an operand read from storage and the original (A). The highest order six bits in the A register are not affected by this operation. The storage address for the operand is contained in index register d.

The logical difference is formed on a bit by bit basis. The logical difference of two binary quantities A and B is defined to be:

A .and..not. B .or. B .and..not. A

Instruction execution begins with a storage reference to index register d. The contents of this index register are read into the X register. A second reference to storage is made using (X) as the storage address. This operand is read into the X register, and the logical difference is then formed and entered into A register. A third storage reference then reads up the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. The storage reference at CP05 may be to either bank depending on the contents of index register d. These two references may occur with all four combinations of possible bank sequences. The timing sequence illustrated below is the shortest of the four possible combinations. Execution time for this case is 15 clock periods. The other three possible combinations result in delays at different points in the sequence, but in each case there are two delays total in the execution of the instruction. The result is that these three cases have an execution time of 25 clock periods. Average execution time for the instruction, assuming equal probability of the four cases, is 22.5 clock periods.

CP00 Transmit (d) to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Set (k) to 1

Read storage bank B to X register

```
        Transmit (X) to S register A

        Begin read/write cycle for storage bank A

        Transmit (X) to Q register
        Set (k) to 2

CP09    Read storage bank A to X register

        Transmit (P) + 1 to S register B

        Begin read/write cycle for storage bank B

        Transmit (X) to Q register
        Transmit (P) + 1 to P register
        Transmit (A) .ld. (X) to A register
        Set (k) to 0

        Read storage bank B to X register
        Read storage bank B to f and d registers
```

44XX Store ((d))

This instruction stores the lowest order 12 bits of (A) in a storage location specified by the contents of index register d. The content of the A register is not altered in this process.

Execution begins with a storage reference to index register d. The contents of this index register are read into the X register. A second reference to storage is made using (X) as the storage address. The data read from storage is discarded in this reference, and the lowest order 12 bits of (A) are recorded. A third storage reference then reads up the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. The storage reference at CP05 may be to either bank depending on the contents of index register d. These two references may occur with all four combinations of possible

bank sequences. The timing sequence illustrated below is the shortest of the four possible combinations. Execution time for this case is 15 clock periods. The other three possible combinations result in delays at different points in the sequence, but in each case there are two delays total in the execution of the instruction. The result is that these three cases have an execution time of 25 clock periods. Average execution time for the instruction, assuming equal probability of the four cases, is 22.5 clock periods.

```
CP00  Transmit (d) to S register B

CP01  Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Set (k) to 1

CP04  Read storage bank B to X register

CP05  Transmit (X) to S register A

CP06  Begin read/write cycle for storage bank A

CP07  Transmit (X) to Q register
      Set (k) to 2

CP09  Read storage bank A to X register

CP10  Transmit (A) to Z register A
      Transmit (P) + 1 to S register B

CP11  Begin read/write cycle for storage bank B

CP12  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

CP14  Read storage bank B to X register
      Read storage bank B to f and d registers
```

45XX Replace add ((d))

This instruction reads an operand from storage and adds it to the current contents of the A register. The result is then left in the A register and is also stored in the same memory location from which the operand was read. The addition is performed in an 18 bit ones complement mode. An 18 bit operand is formed from the 12 bit storage operand by adding six higher order zero bits. The result returned to storage is the lowest order 12 bits of the final (A). The storage address for reading the operand and storing the result is contained in index register d. Note that the result stored is not necessarily equal to the result left in the A register.

There are four storage references required in the execution of this instruction. The first reference reads the contents of index register d into the X register and then into the Q register. A second storage reference is made using (X) as the storage address. This operand is read into the X register and then added to (A). A third storage reference stores the lowest order 12 bits of the resulting sum using (Q) as the storage address. The fourth storage reference reads up the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. The storage reference at CP05 may be to either bank depending on the contents of index register d. These two references may occur with all four combinations of possible bank sequences. The timing sequence illustrated below is the shortest of the four possible combinations. Execution time for this case is 25 clock periods. The other three possible combinations result in delays at different points in the sequence, but in each case there are two delays total in the execution of the instruction. The result is that these three cases have an execution time of 35 clock periods. Average execution time for this instruction, assuming equal probability of the four cases, is 32.5 clock periods.

```
CP00  Transmit (d) to S register B

CP01  Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Set (k) to 1

CP04  Read storage bank B to X register

CP05  Transmit (X) to S register A

CP06  Begin read/write cycle for storage bank A

CP07  Transmit (X) to Q register
      Set (k) to 2

CP09  Read storage bank A to X register

CP15  Transmit (Q) to S register A

CP16  Begin read/write cycle for storage bank A

CP17  Transmit (A) + (X) to A register
      Set (k) to 3

CP19  Read storage bank A to X register

CP20  Transmit (A) to Z register A
      Transmit (P) + 1 to S register B

CP21  Begin read/write cycle for storage bank B

CP22  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

CP24  Read storage bank B to X register
      Read storage bank B to f and d registers
```

46XX Replace add one ((d))

This instruction reads an operand from storage, increases its value by one count, and returns the result to the same storage location. The storage address for reading the operand and storing the result is contained in index register d. The result is left in the A register as well as in storage.

Execution begins by clearing the A register and entering a value of plus one. The operand is then read from storage and added to (A) in an 18 bit ones complement mode. The operand is treated as a 12 bit positive quantity in this process. An 18 bit operand is formed from the 12 bit storage operand by adding six higher order zero bits. The result is left in the A register, and the lowest order 12 bits are returned to storage. Note that the arithmetic is essentially twos complement as viewed from storage, and the quantity in the A register is not necessarily equal to the result in storage.

There are four storage references required in the execution of this instruction. The first reference reads the contents of index register d into the X register and then into the Q register. A second storage reference is made using (X) as the storage address. This operand is read into the X register and then added to (A). A third storage reference stores the lowest order 12 bits of the resulting sum using (Q) as the storage address. The fourth storage reference reads up the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. The storage reference at CP05 may be to either bank depending on the contents of index register d. These two references may occur with all four combinations of possible bank sequences. The timing sequence illustrated below is the shortest of the four possible combinations. Execution time for this case is 25 clock periods. The other three possible combinations result in delays at different points in the sequence, but in each case there are two delays total in the execution of the instruction. The result is that these three cases have an execution time of 35 clock periods. Average execution time for this instruction, assuming equal probability of the four cases, is 32.5 clock periods.

```
CP00  Transmit (d) to S register B

CP01  Begin read/write cycle for storage bank B

CP02  Transmit +1 to A register
      Transmit (d) to Q register
      Set (k) to 1

CP04  Read storage bank B to X register

CP05  Transmit (X) to S register A

CP06  Begin read/write cycle for storage bank A

CP07  Transmit (X) to Q register
      Set (k) to 2

CP09  Read storage bank A to X register

CP15  Transmit (Q) to S register A

CP16  Begin read/write cycle for storage bank A

CP17  Transmit (A) + (X) to A register
      Set (k) to 3

CP19  Read storage bank A to X register

CP20  Transmit (A) to Z register A
      Transmit (P) + 1 to S register B

CP21  Begin read/write cycle for storage bank B

CP22  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

CP24  Read storage bank B to X register
      Read storage bank B to f and d registers
```

47XX Replace subtract one ((d))

This instruction reads an operand from storage, decreases its value by one count, and returns the result to the same storage location. The storage address for reading the operand and storing the result is contained in index register d. The result is left in the A register as well as in storage.

Execution begins by clearing the A register and entering a value of minus one. The operand is then read from storage and added to (A) in an 18 bit ones complement mode. The operand is treated as a 12 bit positive quantity in this process. An 18 bit operand is formed from the 12 bit storage operand by adding six higher order zero bits. The result is left in the A register, and the lowest order 12 bits are returned to storage. Note that the arithmetic is essentially twos complement as viewed from storage, and the quantity in the A register is not necessarily equal to the result in storage.

There are four storage references required in the execution of this instruction. The first reference reads the contents of index register d into the X register and then into the Q register. A second storage reference is made using (X) as the storage address. This operand is read into the X register and then added to (A). A third storage reference stores the lowest order 12 bits of the resulting sum using (Q) as the storage address. The fourth storage reference reads up the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 may be to either bank depending on the value of d. The storage reference at CP05 may be to either bank depending on the contents of index register d. These two references may occur with all four combinations of possible bank sequences. The timing sequence illustrated below is the shortest of the four possible combinations. Execution time for this case is 25 clock periods. The other three possible combinations result in delays at different points in the sequence, but in each case there are two delays total in the execution of the instruction. The result is that these three cases have an execution time of 35 clock periods. Average execution time for the instruction, assuming equal probability of the four cases, is 32.5 clock periods.

```
CP00  Transmit (d) to S register B

CP01  Begin read/write cycle for storage bank B

CP02  Transmit -1 to A register
      Transmit (d) to Q register
      Set (k) to 1

CP04  Read storage bank B to X register

CP05  Transmit (X) to S register A

CP06  Begin read/write cycle for storage bank A

CP07  Transmit (X) to Q register
      Set (k) to 2

CP09  Read storage bank A to X register

CP15  Transmit (Q) to S register A

CP16  Begin read/write cycle for storage bank A

CP17  Transmit (A) + (X) to A register
      Set (k) to 3

CP19  Read storage bank A to X register

CP20  Transmit (A) to Z register A
      Transmit (P) + 1 to S register B

CP21  Begin read/write cycle for storage bank B

CP22  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

CP24  Read storage bank B to X register
      Read storage bank B to f and d registers
```

5000 XXXX Load (m)

This instruction clears the A register and enters a 12 bit operand from storage. The address for the operand is contained in the m designator for this instruction. The operand is entered in the A register as a 12 bit positive integer. The highest order six bits in the A register are always cleared.

Instruction execution begins with a storage reference for the m designator. This quantity is read into the X register. A second storage reference is then made using (X) as the storage address. This operand is read into the X register and then entered in the A register. A third storage reference is made to read the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for the second word of the current instruction. The storage reference at CP05 may be to either bank depending on the value of the m designator. If this reference is to bank B a five clock period delay will occur in the transmission of (Q) + (X) to S register. However, in this case no delay will occur in the storage reference for the next instruction. If the storage reference at CP05 is to bank A no delay will occur in the transmission of (Q) + (X) to S register. A five clock period delay will then occur in the transmission of (P) + 1 to S register for the next instruction word. This second situation is illustrated in the timing sequence below. As a result of these two possible timing sequences the execution time for this instruction is a constant 20 clock periods.

CP00 Transmit (P) + 1 to S register B

Begin read/write cycle for storage bank B

Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 2

Read storage bank B to X register

```
CP05  Transmit (Q) + (X) to S register A

      Begin read/write cycle for storage bank A

CP07  Transmit (Q) + (X) to Q register
      Set (k) to 3

      Read storage bank A to X register

      Transmit (P) + 1 to S register A

CP16  Begin read/write cycle for storage bank A

CP17  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Transmit (X) to A register
      Set (k) to 0

CP19  Read storage bank A to X register
      Read storage bank A to f and d registers
```

50XX XXXX	Load (m + (d))

This instruction clears the A register and enters a 12 bit operand from storage. The address for the operand is formed by adding the m designator and the contents of index register d in a 12 bit ones complement mode. The operand is entered in the A register as a 12 bit positive integer. The highest order six bits in the A register are always cleared. The d designator must have a nonzero value for this instruction.

There are four storage references required in the execution of this instruction. The first reference reads the m designator into the X register and then into the Q register. The second reference reads the contents of index register d into the X register. The third reference uses (Q) + (X) as a storage address for the operand. This quantity is read into the X register and then entered in the A register. The fourth storage reference reads the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for reading the m designator. The storage reference at CP05 may be to either bank depending on the value of the d designator. The storage reference at CP10 may be to either bank depending on the address for the operand. These two references may occur in all four combinations of possible sequences. The timing sequence listed below illustrates the shortest of the four possibilities and results in an execution time of 20 clock periods. This occurs when the reference to the index register is in bank A and the reference for the operand is in bank B. In each of the other three possible sequences there are two delays of five clock periods each introduced by bank conflicts. These cases result in an instruction execution time of 30 clock periods. Average execution time for this instruction assuming equal probability of occurrence for the four cases is 27.5 clock periods.

```
CP00  Transmit (P) + 1 to S register B

CP01  Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 1

      Read storage bank B to X register

      Transmit (Q) to S register A

CP06  Begin read/write cycle for storage bank A

      Transmit (X) to Q register
      Set (k) to 2

      Read storage bank A to X register

      Transmit (Q) + (X) to S register B

      Begin read/write cycle for storage bank B

CP12  Transmit (Q) + (X) to Q register
      Set (k) to 3

      Read storage bank B to X register
```

CP15 Transmit (P) + 1 to S register A

CP16 Begin read/write cycle for storage bank A

CP17 Transmit (X) to Q register
Transmit (P) + 1 to P register
Transmit (X) to A register
Set (k) to 0

CP19 Read storage bank A to X register
Read storage bank A to f and d registers

5100 XXXX	Add (m)

This instruction reads an operand from storage and adds it to the current contents of the A register. The addition is performed in an 18 bit ones complement mode. An 18 bit operand is formed from the 12 bit storage operand by adding six higher order zero bits. The storage address for the operand is contained in the m designator for this instruction.

Instruction execution begins with a storage reference for the m designator. This quantity is read into the X register. A second storage reference is then made using (X) as the storage address. This operand is read into the X register and then entered in the A register. A third storage reference is made to read the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for the second word of the current instruction. The storage reference at CP05 may be to either bank depending on the value of the m designator. If this reference is to bank B a five clock period delay will occur in the transmission of (Q) + (X) to S register. However, in this case no delay will occur in the storage reference for the next instruction. If the storage reference at CP05 is to bank A no delay will occur in the transmission of (Q) + (X) to S register. A five clock period delay will then occur in the transmission of (P) + 1 to S register for the next instruction word. This second situation is illustrated in the timing sequence below. As a result of these two possible timing sequences the execution time for this instruction is a constant 20 clock periods.

```
        Transmit (P) + 1 to S register B

        Begin read/write cycle for storage bank B

        Transmit (d) to Q register
        Transmit (P) + 1 to P register
        Set (k) to 2

CP04    Read storage bank B to X register

        Transmit (Q) + (X) to S register A

        Begin read/write cycle for storage bank A

CP07    Transmit (Q) + (X) to Q register
        Set (k) to 3

CP09    Read storage bank A to X register

CP15    Transmit (P) + 1 to S register A

CP16    Begin read/write cycle for storage bank A

CP17    Transmit (X) to Q register
        Transmit (P) + 1 to P register
        Transmit (A) + (X) to A register
        Set (k) to 0

CP19    Read storage bank A to X register
        Read storage bank A to f and d registers
```

51XX XXXX Add (m + (d))

This instruction reads an operand from storage and adds it to the current contents of the A register. The addition is performed in an 18 bit ones complement mode. An 18 bit operand is formed from the 12 bit storage operand by adding six higher order zero bits. The storage address for the operand is formed by adding the m designator and the contents of index register d in a 12 bit ones complement mode. The d designator must have a nonzero value for this instruction.

There are four storage references required in the execution of this instruction. The first reference reads the m designator into the X register and then into the Q register. The second reference reads the contents of index register d into the X register. The third reference uses (Q) + (X) as a storage address for the operand. This quantity is read into the X register and then entered in the A register. The fourth storage reference reads the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for reading the m designator. The storage reference at CP05 may be to either bank depending on the value of the d designator. The storage reference at CP10 may be to either bank depending on the address for the operand. These two references may occur in all four combinations of possible sequences. The timing sequence listed below illustrates the shortest of the four possibilities and results in an execution time of 20 clock periods. This occurs when the reference to the index register is in bank A and the reference for the operand is in bank B. In each of the other three possible sequences there are two delays of five clock periods each introduced by bank conflicts. These cases result in an instruction execution time of 30 clock periods. Average execution time for this instruction assuming equal probability of occurrence for the four cases is 27.5 clock periods.

```
        Transmit (P) + 1 to S register B

        Begin read/write cycle for storage bank B

CP02    Transmit (d) to Q register
        Transmit (P) + 1 to P register
        Set (k) to 1

CP04    Read storage bank B to X register

CP05    Transmit (Q) to S register A

CP06    Begin read/write cycle for storage bank A

CP07    Transmit (X) to Q register
        Set (k) to 2

CP09    Read storage bank A to X register

CP10    Transmit (Q) + (X) to S register B

CP11    Begin read/write cycle for storage bank B

CP12    Transmit (Q) + (X) to Q register
        Set (k) to 3

CP14    Read storage bank B to X register

CP15    Transmit (P) + 1 to S register A

CP16    Begin read/write cycle for storage bank A

CP17    Transmit (X) to Q register
        Transmit (P) + 1 to P register
        Transmit (A) + (X) to A register
        Set (k) to 0

CP19    Read storage bank A to X register
        Read storage bank A to f and d registers
```

5200 XXXX Subtract (m)

This instruction reads an operand from storage and subtracts it from the current contents of the A register. The result is left in the A register. The storage address for the operand is contained in the m designator for this instruction. The operation is performed by adding the complement of the operand to (A) in an 18 bit ones complement mode. An 18 bit operand for the addition is formed from the 12 bit storage operand by forcing the highest order six bits to a one value. The lowest order 12 bits are the bit by bit complement of the storage operand values.

Instruction execution begins with a storage reference for the m designator. This quantity is read into the X register. A second storage reference is then made using (X) as the storage address. This operand is read into the X register and then subtracted in the A register. A third storage reference is made to read the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for the second word of the current instruction. The storage reference at CP05 may be to either bank depending on the value of the m designator. If this reference is to bank B a five clock period delay will occur in the transmission of (Q) + (X) to S register. However, in this case no delay will occur in the storage reference for the next instruction. If the storage reference at CP05 is to bank A no delay will occur in the transmission of (Q) + (X) to S register. A five clock period delay will then occur in the transmission of (P) + 1 to S register for the next instruction word. This second situation is illustrated in the timing sequence below. As a result of these two possible timing sequences the execution time for this instruction is a constant 20 clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 2

Read storage bank B to X register

```
        Transmit (Q) + (X) to S register A

        Begin read/write cycle for storage bank A

        Transmit (Q) + (X) to Q register
        Set (k) to 3

CP09    Read storage bank A to X register

        Transmit (P) + 1 to S register A

        Begin read/write cycle for storage bank A

        Transmit (X) to Q register
        Transmit (P) + 1 to P register
        Transmit (A) - (X) to A register
        Set (k) to 0

CP19    Read storage bank A to X register
        Read storage bank A to f and d registers
```

52XX XXXX	Subtract (m + (d))

This instruction reads an operand from storage and subtracts it from the current contents of the A register. The result is left in the A register. The address for the operand is formed by adding the m designator and the contents of index register d in a 12 bit ones complement mode. The arithmetic operation is performed by adding the complement of the operand to (A) in an 18 bit ones complement mode. An 18 bit operand for the addition is formed from the 12 bit storage operand by forcing the highest order six bits to a value of one. The lowest order 12 bits are the bit by bit complement of the storage operand values. The d designator must have a nonzero value for this instruction.

There are four storage references required in the execution of this instruction. The first reference reads the m designator into the X register and then into the Q register. The second reference reads the contents of index register d into the X register. The third reference uses (Q) + (X) as a storage address for the operand. This quantity is read into the X register and then subtracted in the A register. The fourth storage reference reads the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for reading the m designator. The storage reference at CP05 may be to either bank depending on the value of the d designator. The storage reference at CP10 may be to either bank depending on the address for the operand. These two references may occur in all four combinations of possible sequences. The timing sequence listed below illustrates the shortest of the four possibilities and results in an execution time of 20 clock periods. This occurs when the reference to the index register is in bank A and the reference for the operand is in bank B. In each of the other three possible sequences there are two delays of five clock periods each introduced by bank conflicts. These cases result in an instruction execution time of 30 clock periods. Average execution time for this instruction assuming equal probability of occurrence for the four cases is 27.5 clock periods.

CP00 Transmit (P) + 1 to S register B

Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 1

CP04 Read storage bank B to X register

CP05 Transmit (Q) to S register A

Begin read/write cycle for storage bank A

Transmit (X) to Q register
Set (k) to 2

CP09 Read storage bank A to X register

CP10 Transmit (Q) + (X) to S register B

CP11 Begin read/write cycle for storage bank B

Transmit (Q) + (X) to Q register
Set (k) to 3

CP14 Read storage bank B to X register

CP15 Transmit (P) + 1 to S register A

CP16 Begin read/write cycle for storage bank A

CP17 Transmit (X) to Q register
Transmit (P) + 1 to P register
Transmit (A) - (X) to A register
Set (k) to 0

CP19 Read storage bank A to X register
Read storage bank A to f and d registers

5300 XXXX	Logical difference (m)

This instruction forms in the A register the logical difference of an operand read from storage and the original (A). The highest order six bits in the A register are not affected by this operation. The storage address for the operand is contained in the m designator for this instruction.

The logical difference is formed on a bit by bit basis. The logical difference of two binary quantities A and B is defined to be:

A .and..not. B .or. B .and..not. A

Instruction execution begins with a storage reference for the m designator. This quantity is read into the X register. A second storage reference is then made using (X) as the storage address. This operand is read into the X register and the logical difference entered in the A register. A third storage reference is made to read the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for the second word of the current instruction. The storage reference at CP05 may be to either bank depending on the value of the m designator. If this reference is to bank B a five clock period delay will occur in the transmission of (Q) + (X) to S register. However, in this case no delay will occur in the storage reference for the next instruction. If the storage reference at CP05 is to bank A no delay will occur in the transmission of (Q) + (X) to S register. A five clock period delay will then occur in the transmission of (P) + 1 to S register for the next instruction word. This second situation is illustrated in the timing sequence below. As a result of these two possible timing sequences the execution time for this instruction is a constant 20 clock periods.

```
      Transmit (P) + 1 to S register B

CP01  Begin read/write cycle for storage bank B

      Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 2

CP04  Read storage bank B to X register

CP05  Transmit (Q) + (X) to S register A

CP06  Begin read/write cycle for storage bank A

CP07  Transmit (Q) + (X) to Q register
      Set (k) to 3

CP09  Read storage bank A to X register

CP15  Transmit (P) + 1 to S register A

CP16  Begin read/write cycle for storage bank A

CP17  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Transmit (A) .ld. (X) to A register
      Set (k) to 0

CP19  Read storage bank A to X register
      Read storage bank A to f and d registers
```

53XX XXXX Logical difference (m + (d))

This instruction forms in the A register the logical difference of an operand read from storage and the original (A). The highest order six bits in the A register are not affected by this operation. The address for the operand is formed by adding the m designator and the contents of index register d in a 12 bit ones complement mode. The d designator must have a nonzero value for this instruction.

The logical difference is formed on a bit by bit basis. The logical difference of two binary quantities A and B is defined to be:

A .and..not. B .or. B .and..not. A

There are four storage references required in the execution of this instruction. The first reference reads the m designator into the X register and then into the Q register. The second reference reads the contents of index register d into the X register. The third reference uses (Q) + (X) as a storage address for the operand. This quantity is read into the X register and the logical difference entered in the A register. The fourth storage reference reads the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for reading the m designator. The storage reference at CP05 may be to either bank depending on the value of the d designator. The storage reference at CP10 may be to either bank depending on the address for the operand. These two references may occur in all four combinations of possible sequences. The timing sequence listed below illustrates the shortest of the four possibilities and results in an execution time of 20 clock periods. This occurs when the reference to the index register is in bank A and the reference for the operand is in bank B. In each of the other three possible sequences there are two delays of five clock periods each introduced by bank conflicts. These cases result in an instruction execution time of 30 clock periods. Average execution time for this instruction assuming equal probability of occurrence for the four cases is 27.5 clock periods.

```
CP00  Transmit (P) + 1 to S register B

CP01  Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 1

CP04  Read storage bank B to X register

CP05  Transmit (Q) to S register A

CP06  Begin read/write cycle for storage bank A

CP07  Transmit (X) to Q register
      Set (k) to 2

CP09  Read storage bank A to X register

CP10  Transmit (Q) + (X) to S register B

CP11  Begin read/write cycle for storage bank B

CP12  Transmit (Q) + (X) to Q register
      Set (k) to 3

CP14  Read storage bank B to X register

CP15  Transmit (P) + 1 to S register A

CP16  Begin read/write cycle for storage bank A

CP17  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Transmit (A) .ld. (X) to A register
      Set (k) to 0

CP19  Read storage bank A to X register
      Read storage bank A to f and d registers
```

5400 XXXX Store (m)

This instruction stores the lowest order 12 bits of (A) in a storage location specified by the m designator. The content of the A register is not altered in this process.

Execution begins with a storage reference for the m designator. This quantity is read into the X register. A second storage reference is made using (X) as the storage address. The lowest order 12 bits of (A) are stored in the write portion of this storage cycle. A third storage reference is then made to read the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for the second word of the current instruction. The storage reference at CP05 may be to either bank depending on the value of the m designator. If this reference is to bank B a five clock period delay will occur in the transmission of (Q) + (X) to S register. However, in this case no delay will occur in the storage reference for the next instruction. If the storage reference at CP05 is to bank A no delay will occur in the transmission of (Q) + (X) to S register. A five clock period delay will then occur in the transmission of (P) + 1 to S register for the next instruction word. This second situation is illustrated in the timing sequence below. As a result of these two possible timing sequences the execution time for this instruction is a constant 20 clock periods.

Transmit (P) + 1 to S register B

Begin read/write cycle for storage bank B

Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 2

CP04 Read storage bank B to X register

CP05 Transmit (Q) + (X) to S register A

CP06 Begin read/write cycle for storage bank A

Transmit (Q) + (X) to Q register
Set (k) to 3

CP09 Read storage bank A to X register

CP10 Transmit (A) to Z register A

CP15 Transmit (P) + 1 to S register A

CP16 Begin read/write cycle for storage bank A

CP17 Transmit (X) to Q register
Transmit (P) + 1 to P register
Set (k) to 0

CP19 Read storage bank A to X register
Read storage bank A to f and d registers

54XX XXXX	Store (m + (d))

This instruction stores the lowest order 12 bits of (A). The storage address is formed by adding the m designator and the contents of index register d in a 12 bit ones complement mode. The d designator must have a nonzero value for this instruction.

There are four storage references required in the execution of this instruction. The first reference reads the m designator into the X register and then into the Q register. The second reference reads the contents of index register d into the X register. The third reference uses (Q) + (X) as a storage address for storing the lowest order 12 bits of the A register. The fourth storage reference reads the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for reading the m designator. The storage reference at CP05 may be to either bank depending on the value of the d designator. The storage reference at CP10 may be to either bank depending on the address for the operand. These two references may occur in all four combinations of possible sequences. The timing sequence listed below illustrates the shortest of the four possibilities and results in an execution time of 20 clock periods. This occurs when the reference

to the index register is in bank A and the reference for the operand is in bank B. In each of the other three possible sequences there are two delays of five clock periods each introduced by bank conflicts. These cases result in an instruction execution time of 30 clock periods. Average execution time for this instruction assuming equal probability of occurrence for the four cases is 27.5 clock periods.

```
CP00  Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

      Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 1

      Read storage bank B to X register

      Transmit (Q) to S register A

      Begin read/write cycle for storage bank A

      Transmit (X) to Q register
      Set (k) to 2

CP09  Read storage bank A to X register

CP10  Transmit (Q) + (X) to S register B

CP11  Begin read/write cycle for storage bank B

CP12  Transmit (Q) + (X) to Q register
      Set (k) to 3

      Read storage bank B to X register

CP15  Transmit (P) + 1 to S register A
      Transmit (A) to Z register B

      Begin read/write cycle for storage bank A

      Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

CP19  Read storage bank A to X register
      Read storage bank A to f and d registers
```

5500 XXXX Replace add (m)

This instruction reads an operand from storage and adds it to the current contents of the A register. The result is left in the A register and is also stored in the same memory location from which the operand was read. The addition is performed in an 18 bit ones complement mode. An 18 bit operand is formed from the 12 bit storage operand by adding six higher order zero bits. The result returned to storage is the lowest order 12 bits of the final (A). The storage address for reading the operand and storing the result is contained in the m designator for this instruction. Note that the result stored is not necessarily equal to the result left in the A register.

There are four storage references required in the execution of this instruction. The first reference reads the m designator into the X register and the Q register. A second reference is made using (X) as the storage address. This operand is read into the X register and is added into the A register. A third reference stores the lowest order 12 bits of (A) using (Q) as the storage address. The fourth reference reads up the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for the m designator. The storage reference at CP05 may be to either bank depending on the value of the m designator. If this reference is to bank B a five clock period delay will occur in the transmission of (Q) + (X) to S register. However, in this case no delay will occur in the storage reference for the next instruction. If the storage reference at CP05 is to bank A, no delay will occur at this time but a five clock period delay will occur in the transmission of (P) + 1 to P for the next instruction word. This second situation is illustrated in the timing sequence below. In either case a five clock period delay will occur between the reading of the operand and the storing of the operand in the same storage location. As a result of the compensating delays in the two possible timing sequences the execution time for this instruction is a constant 30 clock periods.

```
CP00  Transmit (P) + 1 to S register B

CP01  Begin read/write cycle for storage bank B

      Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 2

      Read storage bank B to X register

CP05  Transmit (Q) + (X) to S register A

CP06  Begin read/write cycle for storage bank A

CP07  Transmit (Q) + (X) to Q register
      Set (k) to 3

CP09  Read storage bank A to X register

CP15  Transmit (Q) to S register A

CP16  Begin read/write cycle for storage bank A

      Transmit (A) + (X) to A register
      Set (k) to 4

CP19  Read storage bank A to X register

CP20  Transmit (A) to Z register A

CP25  Transmit (P) + 1 to S register A

CP26  Begin read/write cycle for storage bank A

CP27  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

CP29  Read storage bank A to X register
      Read storage bank A to f and d registers
```

55XX XXXX Replace add (m + (d))

This instruction reads an operand from storage and adds it to the current contents of the A register. The result is left in the A register and is also stored in the same memory location from which the operand was read. The addition is performed in an 18 bit ones complement mode. An 18 bit operand is formed from the 12 bit storage operand by adding six higher order zero bits. The result returned to storage is the lowest order 12 bits of the final (A). The storage address for reading the operand and storing the result is formed by adding the m designator to the contents of index register d in a 12 bit ones complement mode. Note that the result stored is not necessarily equal to the result left in the A register.

There are five storage references required in the execution of this instruction. The first reference reads the m designator into the X register and then into the Q register. The second reference reads the contents of index register d into the X register. The third reference uses (Q) + (X) to read the operand into the X register and the addition is performed in the A register. The quantity (Q) + (X) is entered in the Q register at this same time. The fourth storage reference stores the lowest order 12 bits of (A) using the new (Q) as a storage address. The last storage reference reads the next program instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for reading the m designator. The storage reference at CP05 may be to either bank depending on the value of the d designator. The storage reference at CP10 may be to either bank depending on the address of the operand. These two references may occur in all four combinations of possible sequences. The timing sequence listed below illustrates the shortest of the four possibilities and results in an execution time of 30 clock periods. This occurs when the reference to the index register is in bank A and the reference for the operand is in bank B. In each of the other three possible sequences there are two delays of five clock periods each introduced by bank conflicts. These cases result in an instruction execution time of 40 clock periods. Average execution time for this instruction assuming equal probability of occurrence for the four cases is 37.5 clock periods.

```
        Transmit (P) + 1 to S register B

CP01    Begin read/write cycle for storage bank B

CP02    Transmit (d) to Q register
        Transmit (P) + 1 to P register
        Set (k) to 1

CP04    Read storage bank B to X register

CP05    Transmit (Q) to S register A

CP06    Begin read/write cycle for storage bank A

CP07    Transmit (X) to Q register
        Set (k) to 2

        Read storage bank A to X register

CP10    Transmit (Q) + (X) to S register B

CP11    Begin read/write cycle for storage bank B

        Transmit (Q) + (X) to Q register
        Set (k) to 3

        Read storage bank B to X register

        Transmit (Q) to S register B

CP21    Begin read/write cycle for storage bank B

CP22    Transmit (A) + (X) to A register
        Set (k) to 4

CP24    Read storage bank B to X register

CP25    Transmit (A) to Z register B
        Transmit (P) + 1 to S register A

CP26    Begin read/write cycle for storage bank A

CP27    Transmit (X) to Q register
        Transmit (P) + 1 to P register
        Set (k) to 0

CP29    Read storage bank A to X register
        Read storage bank A to f and d registers
```

5600 XXXX Replace add one (m)

This instruction reads an operand from storage, increases its value by one count, and returns the result to the same storage location. The storage address for reading the operand and storing the result is contained in the m designator for this instruction. The result is left in the A register as well as in storage.

Execution begins by clearing the A register and entering a value of plus one. The operand is then read from storage and added to (A) in an 18 bit ones complement mode. The operand is treated as a 12 bit positive quantity in this process. An 18 bit operand is formed from the 12 bit storage operand by adding six higher order zero bits. The result is left in the A register, and the lowest order 12 bits are returned to storage. Note that the arithmetic is essentially twos complement as viewed from storage, and the quantity in the A register is not necessarily equal to the result in storage.

There are four storage references required in the execution of this instruction. The first reference reads the m designator from storage into the X register and then into the Q register. A second storage reference is made using (X) as the storage address. This operand is read into the X register and is added into the A register. A third reference stores the lowest order 12 bits of (A) using (Q) as the storage address. The fourth reference reads up the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for the m designator. The storage reference at CP05 may be to either bank depending on the value of the m designator. If this reference is to bank B a five clock period delay will occur in the transmission of (Q) + (X) to S register. However, in this case no delay will occur in the storage reference for the next instruction. If the storage reference at CP05 is to bank A, no delay will occur at this time but a five clock period delay will occur in the transmission of (P) + 1 to P for the next instruction word. This second situation is illustrated in the timing sequence below. In either case a five clock period delay will occur between the reading of the operand and the storing of the operand in the same storage location. As a result of the compensating delays in the two possible timing sequences the execution time for this instruction is a constant 30 clock periods.

```
CP00  Transmit (P) + 1 to S register B

CP01  Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Transmit +1 to A register
      Set (k) to 2

      Read storage bank B to X register

CP05  Transmit (Q) + (X) to S register A

      Begin read/write cycle for storage bank A

CP07  Transmit (Q) + (X) to Q register
      Set (k) to 3

      Read storage bank A to X register

CP15  Transmit (Q) to S register A

CP16  Begin read/write cycle for storage bank A

CP17  Transmit (A) + (X) to A register
      Set (k) to 4

CP19  Read storage bank A to X register

      Transmit (A) to Z register A

CP25  Transmit (P) + 1 to S register A

CP26  Begin read/write cycle for storage bank A

CP27  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

CP29  Read storage bank A to X register
      Read storage bank A to f and d registers
```

56XX XXXX	Replace add one (m + (d))

This instruction reads an operand from storage, increases its value by one count, and returns the result to the same storage location. The storage address for reading the operand and storing the result is formed by adding the m designator to the contents of index register d in a 12 bit ones complement mode. The result is left in the A register as well as in storage.

Execution begins by clearing the A register and entering a value of plus one. The m designator is read from storage and entered in the X register and then into the Q register. A second storage reference reads the contents of index register d into the X register. A third reference reads the operand into the X register using (Q) + (X) as the storage address. The quantity (Q) + (X) is entered in the Q register at this same time. The operand is then added into the A register in an 18 bit ones complement mode. An 18 bit operand is formed from the 12 bit storage operand by adding six higher order zero bits. The result is left in the A register, and the lowest order 12 bits are returned to storage using (Q) as the storage address. Note that the arithmetic is essentially twos complement as viewed from storage, and the quantity in the A register is not necessarily equal to the result in storage. A fifth storage reference reads up the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for reading the m designator. The storage reference at CP05 may be to either bank depending on the value of the d designator. The storage reference at CP10 may be to either bank depending on the address of the operand. These two references may occur in all four combinations of possible sequences. The timing sequence listed below illustrates the shortest of the four possibilities and results in an execution time of 30 clock periods. This occurs when the reference to the index register is in bank A and the reference for the operand is in bank B. In each of the other three possible sequences there are two delays of five clock periods each introduced by bank conflicts. These cases result in an instruction execution time of 40 clock periods. Average execution time for this instruction assuming equal probability of occurrence for the four cases is 37.5 clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Transmit +1 to A register
Set (k) to 1

CP04 Read storage bank B to X register

CP05 Transmit (Q) to S register A

CP06 Begin read/write cycle for storage bank A

CP07 Transmit (X) to Q register
Set (k) to 2

CP09 Read storage bank A to X register

CP10 Transmit (Q) + (X) to S register B

CP11 Begin read/write cycle for storage bank B

CP12 Transmit (Q) + (X) to Q register
Set (k) to 3

CP14 Read storage bank B to X register

CP20 Transmit (Q) to S register B

CP21 Begin read/write cycle for storage bank B

CP22 Transmit (A) + (X) to A register
Set (k) to 4

CP24 Read storage bank B to X register

CP25 Transmit (A) to Z register B
Transmit (P) + 1 to S register A

CP26 Begin read/write cycle for storage bank A

CP27 Transmit (X) to Q register
Transmit (P) + 1 to P register
Set (k) to 0

CP29 Read storage bank A to X register
Read storage bank A to f and d registers

5700 XXXX Replace subtract one (m)

This instruction reads an operand from storage, decreases its value by one count, and returns the result to the same storage location. The storage address for reading the operand and storing the result is contained in the m designator for this instruction. The result is left in the A register as well as in storage.

Execution begins by clearing the A register and entering a value of minus one. The operand is then read from storage and added to (A) in an 18 bit ones complement mode. The operand is treated as a 12 bit positive quantity in this process. An 18 bit operand is formed from the 12 bit storage operand by adding six higher order zero bits. The result is left in the A register, and the lowest order 12 bits are returned to storage. Note that the arithmetic is essentially twos complement as viewed from storage, and the quantity in the A register is not necessarily equal to the result in storage.

There are four storage references required in the execution of this instruction. The first reference reads the m designator from storage into the X register and then into the Q register. A second storage reference is made using (X) as the storage address. This operand is read into the X register and is added into the A register. A third reference stores the lowest order 12 bits of (A) using (Q) as the storage address. The fourth reference reads up the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for the m designator. The storage reference at CP05 may be to either bank depending on the value of the m designator. If this reference is to bank B a five clock period delay will occur in the transmission of (Q) + (X) to S register. However, in this case no delay will occur in the storage reference for the next instruction. If the storage reference at CP05 is to bank A, no delay will occur at this time but a five clock period delay will occur in the transmission of (P) + 1 to P for the next instruction word. This second situation is illustrated in the timing sequence below. In either case a five clock period delay will occur between the reading of the operand and the storing of the operand in the same storage location. As a result of the compensating delays in the two possible timing sequences the execution time for this instruction is a constant 30 clock periods.

```
CP00  Transmit (P) + 1 to S register B

CP01  Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Transmit -1 to A register
      Set (k) to 2

      Read storage bank B to X register

CP05  Transmit (Q) + (X) to S register A

CP06  Begin read/write cycle for storage bank A

CP07  Transmit (Q) + (X) to Q register
      Set (k) to 3

      Read storage bank A to X register

      Transmit (Q) to S register A

CP16  Begin read/write cycle for storage bank A

CP17  Transmit (A) + (X) to A register
      Set (k) to 4 .

      Read storage bank A to X register

      Transmit (A) to Z register A

CP25  Transmit (P) + 1 to S register A

CP26  Begin read/write cycle for storage bank A

CP27  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

      Read storage bank A to X register
      Read storage bank A to f and d registers
```

57XX XXXX Replace subtract one (m + (d))

This instruction reads an operand from storage, decreases its value by one count, and returns the result to the same storage location. The storage address for reading the operand and storing the result is formed by adding the m designator to the contents of index register d in a 12 bit ones complement mode. The result is left in the A register as well as in storage.

Execution begins by clearing the A register and entering a value of minus one. The m designator is read from storage and entered in the X register and then into the Q register. A second storage reference reads the contents of index register d into the X register. A third reference reads the operand into the X register using (Q) + (X) as the storage address. The quantity (Q) + (X) is entered in the Q register at this same time. The operand is then added into the A register in an 18 bit ones complement mode. An 18 bit operand is formed from the 12 bit storage operand by adding six higher order zero bits. The result is left in the A register, and the lowest order 12 bits are returned to storage using (Q) as the storage address. Note that the arithmetic is essentially twos complement as viewed from storage, and the quantity in the A register is not necessarily equal to the result in storage. A fifth storage reference reads up the next instruction word.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for reading the m designator. The storage reference at CP05 may be to either bank depending on the value of the d designator. The storage reference at CP10 may be to either bank depending on the address of the operand. These two references may occur in all four combinations of possible sequences. The timing sequence listed below illustrates the shortest of the four possibilities and results in an execution time of 30 clock periods. This occurs when the reference to the index register is in bank A and the reference for the operand is in bank B. In each of the other three possible sequences there are two delays of five clock periods each introduced by bank conflicts. These cases result in an instruction execution time of 40 clock periods. Average execution time for this instruction assuming equal probability of occurrence for the four cases is 37.5 clock periods.

```
        Transmit (P) + 1 to S register B

        Begin read/write cycle for storage bank B

CP02    Transmit (d) to Q register
        Transmit (P) + 1 to P register
        Transmit -1 to A register
        Set (k) to 1

CP04    Read storage bank B to X register

CP05    Transmit (Q) to S register A

CP06    Begin read/write cycle for storage bank A

CP07    Transmit (X) to Q register
        Set (k) to 2

CP09    Read storage bank A to X register

CP10    Transmit (Q) + (X) to S register B

CP11    Begin read/write cycle for storage bank B

CP12    Transmit (Q) + (X) to Q register
        Set (k) to 3

        Read storage bank B to X register

        Transmit (Q) to S register B

        Begin read/write cycle for storage bank B

CP22    Transmit (A) + (X) to A register
        Set (k) to 4

CP24    Read storage bank B to X register

CP25    Transmit (A) to Z register B
        Transmit (P) + 1 to S register A

        Begin read/write cycle for storage bank A

CP27    Transmit (X) to Q register
        Transmit (P) + 1 to P register
        Set (k) to 0

CP29    Read storage bank A to X register
        Read storage bank A to f and d registers
```

60XX XXXX Jump on input word flag

This is a conditional branch instruction which continues the current program sequence or jumps to a new program sequence, depending on the condition of the input channel d word flag. A new program sequence is initiated beginning at address m if the input channel d word flag is set. The current program sequence is continued if the input channel d word flag is not set.

Execution time

There are two possible timing sequences for this instruction, each requiring two storage references. The first sequence is listed below and represents the timing for the case where the jump is not taken. The branch decision is based on the IWF status during CP05. The storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 must be to bank B for this case to read the m designator. The storage reference at CP05 must be to bank A to read the next instruction word. Execution time is a constant 10 clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 1

CP04 Read storage bank B to X register

CP05 Transmit (P) + 1 to S register A

CP06 Begin read/write cycle for storage bank A

CP07 Transmit (X) to Q register
Transmit (P) + 1 to P register
Set (k) to 0

CP09 Read storage bank A to X register
Read storage bank A to f and d registers

The second timing sequence is listed below and represents the timing for the case where the jump is taken. The storage reference at CP00 must be to bank B to read the m designator. The storage reference at CP05 may be to either bank depending on the value of the m designator. If this reference is to bank B a five clock period delay will be introduced in the transmission of (P) + 1 to S register. If this reference is to bank A there will be no delay. Execution time will be 10 clock periods or 15 clock periods, depending on the value of the m designator.

```
CP00  Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

      Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 1

      Read storage bank B to X register

      Transmit (X) to S register A

      Begin read/write cycle for storage bank A

      Transmit (P) + 1 to Q register
      Transmit (X) to P register
      Set (k) to 0

CP09  Read storage bank A to X register
      Read storage bank A to f and d registers
```

61XX XXXX Jump on no input word flag

This is a conditional branch instruction which continues the current program sequence or jumps to a new program sequence, depending on the condition of the input channel d word flag. A new program sequence is initiated beginning at address m if the input channel d word flag is not set. The current program sequence is continued if the input channel d word flag is set. Execution timing for this instruction is the same as that listed for the 60 instruction.

62XX XXXX Jump on input record flag

This is a conditional branch instruction which continues the current program sequence or jumps to a new program sequence, depending on the condition of the input channel d record flag. A new program sequence is initiated beginning at address m if the input channel d record flag is set. The current program sequence is continued if the input channel d record flag is not set. Execution timing for this instruction is the same as that listed for the 60 instruction.

63XX XXXX Jump on no input record flag

This is a conditional branch instruction which continues the current program sequence or jumps to a new program sequence, depending on the condition of the input channel d record flag. A new program sequence is initiated beginning at address m if the input channel d record flag is not set. The current program sequence is continued if the input channel d record flag is set. Execution timing for this instruction is the same as that listed for the 60 instruction.

64XX XXXX Jump on output word flag

This is a conditional branch instruction which continues the current program sequence or jumps to a new program sequence, depending on the condition of the output channel d word flag. A new program sequence is initiated beginning at address m if the output channel d word flag is set. The current program sequence is continued if the output channel d word flag is not set. Execution timing for this instruction is the same as that listed for the 60 instruction.

65XX XXXX Jump on no output word flag

This is a conditional branch instruction which continues the current program sequence or jumps to a new program sequence, depending on the condition of the output channel d word flag. A new program sequence is initiated beginning at address m if the output channel d word flag is not set. The current program sequence is continued if the output channel d word flag is set. Execution timing for this instruction is the same as that listed for the 60 instruction.

66XX XXXX Jump on output record flag

This is a conditional branch instruction which continues the current program sequence or jumps to a new program sequence, depending on the condition of the output channel d record flag. A new program sequence is initiated beginning at address m if the output channel d record flag is set. The current program sequence is continued if the output channel d record flag is not set. Execution timing for this instruction is the same as that listed for the 60 instruction.

67XX XXXX Jump on no output record flag

This is a conditional branch instruction which continues the current program sequence or jumps to a new program sequence, depending on the condition of the output channel d record flag. A new program sequence is initiated beginning at address m if the output channel d record flag is not set. The current program sequence is continued if the output channel d record flag is set. Execution timing for this instruction is the same as that listed for the 60 instruction.

70XX Input to A from channel d

This instruction reads one word from input channel d and enters the word in the A register. This instruction will not be executed until the input channel d word flag is set. If the flag is not set at the time the instruction is read from storage the PPU program will stop with the instruction in the f and d registers and wait until the flag is set by an external signal. The input channel d record flag does not affect execution of this instruction. This instruction clears the input channel d word flag and record flag and transmits a resume signal over the input cable after the word has been read into the A register.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The transmission of the instruction code from storage bank A to the f and d registers is assumed to occur in the clock period before CP00 in the timing chart. CP00 is then the first clock period in which the instruction translation exists. The timing chart assumes that the input channel d word flag is also set during CP00. If this is true the execution time for the instruction is nine clock periods. If this were not true the execution of the remainder of the instruction would be delayed by the amount of time required for the input channel d word flag to set. Synchronizing of the input channel d word flag status into the IWF occurs during the first four clock periods in the instruction execution.

CP00 Begin synchronizing function

Set IWF

CP04 Transmit (P) + 1 to S register B

CP05 Begin read/write cycle for storage bank B

Transmit (d) to Q register
Transmit (P) + 1 to P register
Transmit channel data to A register
Set input channel d resume flag
Clear input channel d word flag and record flag
Set (k) to 0

Transmit resume pulse over input channel d cable

Read storage bank B to X register
Read storage bank B to f and d registers

71XX XXXX	Input (A) words to m from channel d

This instruction reads a block of data arriving on input channel d and stores the data in consecutive address locations in storage. The initial storage location for the block is specified by the m designator. The length of the block is specified by the initial contents of the A register or by a record flag on the input channel.

Instruction execution begins with a storage reference for the m designator. This quantity is read into the X register and then entered in the Q register. The Q register now contains the address for the first word of the data block. The d register contains the channel number, and the A register contains a word count for the block. If (A) is zero at this time the instruction sequence is terminated and the next instruction word is read from storage.

The input channel d word flag must be set before the first word of the block can be entered in storage. If this flag is not set when the instruction is initiated the PPU program will stop with the instruction in the f and d registers and wait until the flag is set by an external signal. The presence of an input channel d record flag is ignored for the first word of the block.

When the input channel d word flag is set the word on the input channel data lines is read into PPU storage at location (Q). The content of the A register is reduced by one count. The content of the Q register is increased by one count in a 12 bit ones complement mode. The input channel d word flag and input channel d record flag are cleared, and a resume pulse is transmitted over the input cable. If the content of the A register is now zero the instruction sequence is terminated and the next instruction word is read from storage. If (A) is not zero the PPU program waits for the setting of the input channel d word flag for the next word of the block.

The setting of the input channel d record flag terminates the block input at any word after the first word. In this case the sequence is terminated with (A) decremented by the number of words actually transmitted over the input channel. A "noise" word is entered in the next sequential storage location in the PPU block input storage area. The remaining locations in the PPU storage area are unaltered.

Execution time

Several timing sequences for this instruction are listed below to illustrate the different situations which can terminate the block input of data. In each case the storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 in each case is to bank B to read the second word for the current instruction. The last storage reference in each case must be to bank A to read the first word of the next instruction.

The first timing chart illustrates the case where a block input is terminated by a record flag rather than exhausting the count in the A register. In this case the external device transmitting the data over the input channel is assumed to have a two clock period response from resume to word flag. This is a very fast response and is probably not typical of a word by word device. For a device with a longer response time the block transfer will be lengthened by the exact amount of the additional delay. A three word block followed by a record flag is used in the illustration. Total execution time is 42 clock periods assuming that the channel d input word flag is set at the time of instruction initiation and the first data reference is to bank A.

```
      Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 1

CP04  Read storage bank B to X register

      Transmit (X) to S register A

CP06  Begin read/write cycle for storage bank A

      Transmit (X) to Q register
      Transmit (A) - 1 to A register
      Set input channel d resume flag
      Clear input channel d word flag and record flag
      Set (k) to 2

CP08  Transmit resume pulse over input channel d cable
```

CP09 Read storage bank A to X register

CP10 Transmit channel data to Z register A
Set input channel d word flag

CP13 Set IWF

CP14 Transmit (Q) + 1 to S register B

CP15 Begin read/write cycle for storage bank B

CP16 Transmit (Q) + 1 to Q register
Transmit (A) - 1 to A register
Set input channel d resume flag
Clear input channel d word flag and record flag
Set (k) to 2

CP17 Transmit resume pulse over input channel d cable

CP18 Read storage bank B to X register

CP19 Transmit channel data to Z register B
Set input channel d word flag

CP22 Set IWF

CP23 Transmit (Q) + 1 to S register A

CP24 Begin read/write cycle for storage bank A

CP25 Transmit (Q) + 1 to Q register
Transmit (A) - 1 to A register
Set input channel d resume flag
Clear input channel d word flag and record flag
Set (k) to 2

CP26 Transmit resume pulse over input channel d

CP27 Read storage bank A to X register

CP28 Transmit channel data to Z register A
Set input channel d record flag

CP31 Set IRF

CP32 Transmit (Q) + 1 to S register B

```
      Begin read/write cycle for storage bank B

CP34  Transmit (Q) + 1 to Q register
      Set (k) to 3

CP36  Read storage bank B to X register

CP37  Transmit (P) + 1 to S register A
      Transmit channel data to Z register B (noise)

CP38  Begin read/write cycle for storage bank A

CP39  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

CP41  Read storage bank A to X register
      Read storage bank A to f and d registers
```

The second timing chart illustrates the case where a block input is terminated by exhausting the count in the A register. This case assumes the same transmitting device as the first example but assumes a count of two in the A register. Total execution time is 24 clock periods assuming that the channel d input word flag is set at the time of instruction initiation and the first data reference is to bank A.

```
CP00  Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 1

CP04  Read storage bank B to X register

CP05  Transmit (X) to S register A

CP06  Begin read/write cycle for storage bank A
```

CP07 Transmit (X) to Q register
Transmit (A) - 1 to A register
Set input channel d resume flag
Clear input channel d word flag and record flag
Set (k) to 2

CP08 Transmit resume pulse over input channel d cable

CP09 Read storage bank A to X register

CP10 Transmit channel data to Z register A
Set input channel d word flag

CP13 Set IWF

CP14 Transmit (Q) + 1 to S register B

CP15 Begin read/write cycle for storage bank B

CP16 Transmit (Q) + 1 to Q register
Transmit (A) - 1 to A register
Set input channel d resume flag
Clear input channel d word flag and record flag
Set (k) to 2

CP17 Transmit resume pulse over input channel d cable

CP18 Read storage bank B to X register

CP19 Transmit (P) + 1 to S register A
Transmit channel data to Z register B

CP20 Begin read/write cycle for storage bank A

CP21 Transmit (X) to Q register
Transmit (P) + 1 to P register
Set (k) to 0

CP23 Read storage bank A to X register
Read storage bank A to f and d registers

The third timing chart illustrates the case where a block input instruction is initiated with a zero value in the A register. The execution time for this case is a constant 10 clock periods, and the input channel d word flag need not be set for instruction execution.

```
CP00  Transmit (P) + 1 to S register B

CP01  Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 1

CP04  Read storage bank B to X register

CP05  Transmit (P) + 1 to S register A

CP06  Begin read/write cycle for storage bank A

CP07  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

CP09  Read storage bank A to X register
      Read storage bank A to f and d registers
```

72XX Output from A on channel d

This instruction transmits one word over output channel d from the lowest order 12 bits of (A). The A register content is not altered in the process. This instruction will not be executed while the output channel d word flag is set. If the flag is set from a previous output instruction the PPU program will stop with this instruction in the f and d registers and wait for an external resume signal to clear the output channel d word flag. When this instruction is executed the output channel d word flag is set and a word pulse is transmitted over the output channel d cable.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The transmission of the instruction word from storage bank A to the f and d registers is assumed to occur in the clock period before CP00 in the timing chart. CP00 is then the first clock period in which the instruction translation exists. The timing chart assumes that the output channel d word flag is cleared during CP00. If this is true the execution time for the instruction is nine clock periods. If this is not true the execution of the remainder of the instruction will be delayed by the amount of time which elapses before the output channel d word flag is cleared. Synchronization of the output channel d word flag into the OWF occurs during the first four clock periods of the instruction execution.

```
      Begin synchronizing function

      Clear OWF

      Transmit (P) + 1 to S register B

      Begin read/write cycle for storage bank B

      Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Transmit (A) to channel d output register
      Set (k) to 0

      Set output channel d word flag
      Transmit word pulse over output channel d cable

CP08  Read storage bank B to X register
      Read storage bank B to f and d registers
```

73XX XXXX Output (A) words from m on channel d

This instruction transmits a block of data over output channel d from consecutive storage locations beginning at address m. The length of the block is specified by the initial contents of the A register. A zero length will cause the instruction to be executed as a pass instruction.

Instruction execution begins with a storage reference for the m designator. This quantity is read into the X register and then entered in the Q register. The Q register now contains the address for the first word of the data block. The d register contains the channel number, and the A register contains the word count for the block. If (A) is zero at this time the instruction sequence is terminated and the next instruction word is read from storage.

The output channel d word flag must be cleared before the first word of the block can be transmitted over the channel. If this flag is set when the instruction is initiated the PPU program will stop with the instruction in the f and d registers and wait until the flag is cleared by a resume pulse over the output channel d cable. The presence of an output channel d record flag has no effect on the execution of this instruction.

When the output channel d word flag is cleared a word is read from storage location (Q) and is entered in the channel d output register. The output channel d word flag is set, and a word pulse is transmitted over the output cable. The content of the A register is reduced by one count. The content of the Q register is increased by one count in a 12 bit ones complement mode. If the content of the A register is now zero the instruction is terminated and the next instruction is read from storage. If (A) is not zero the PPU program waits for the output channel d word flag to clear and repeats the sequence for the next word of the block.

Execution time

Two timing sequences are listed below for this instruction. The first sequence illustrates the timing for a three word output block. The second sequence illustrates the timing for the case of zero words. In each case the storage banks are designated as bank A and bank B where bank A contains the first word of the current instruction. The storage reference at CP00 in each case is to bank B for the second word of the current instruction. The last storage reference must be to bank A to read the first word of the next instruction.

This timing sequence illustrates the execution of the 73 instruction with an initial (A) = 3. The external device receiving the data is assumed to have a two clock period response from receipt of word pulse to transmission of resume pulse. For a device with a longer response time the block transfer will be lengthened by a corresponding amount. Total execution time for this example is 34 clock periods, assuming that the output channel d word flag is cleared when the instruction begins and the first word of the block is read from bank A.

```
CP00  Transmit (P) + 1 to S register B

CP01  Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 1

      Read storage bank B to X register

      Transmit (X) to S register A

      Begin read/write cycle for storage bank A

      Transmit (X) to Q register
      Transmit (A) - 1 to A register
      Set (k) to 2

CP09  Read storage bank A to X register

CP10  Transmit (Q) + 1 to S register B

      Begin read/write cycle for storage bank B

CP12  Transmit (Q) + 1 to Q register
      Transmit (A) - 1 to A register
      Transmit (X) to channel d output register
      Set output channel d word flag
      Set (k) to 2

CP13  Transmit word pulse over output channel d cable

      Read storage bank B to X register

      Receive resume pulse over output channel d cable
      Clear output channel d word flag and record flag
```

```
CP18  Clear OWF

CP19  Transmit (Q) + 1 to S register A

CP20  Begin read/write cycle for storage bank A

CP21  Transmit (Q) + 1 to Q register
      Transmit (A) - 1 to A register
      Transmit (X) to channel d output register
      Set output channel d word flag
      Set (k) to 2

CP22  Transmit word pulse over output channel d cable
      Set OWF

CP23  Read storage bank A to X register

CP24  Receive resume pulse over output channel d cable
      Clear output channel d word flag

CP27  Clear OWF

CP29  Transmit (P) + 1 to S register A

CP30  Begin read/write cycle for storage bank A.

CP31  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

CP33  Read storage bank A to X register
      Read storage bank A to f and d registers
```

The following timing sequence illustrates the execution of the 73 instruction with an initial (A) = 0. In this case no data is transmitted over the channel d output cable. Execution time for this case is a constant 10 clock periods.

```
CP00  Transmit (P) + 1 to S register B

CP01  Begin read/write cycle for storage bank B

CP02  Transmit (d) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 1

CP04  Read storage bank B to X register

CP05  Transmit (P) + 1 to S register A

CP06  Begin read/write cycle for storage bank A

CP07  Transmit (X) to Q register
      Transmit (P) + 1 to P register
      Set (k) to 0

CP09  Read storage bank A to X register
      Read storage bank A to f and d registers
```

74XX Output record flag on channel d

This instruction sets the output channel d record flag and transmits a record pulse over the output channel d cable. The previous status of the output channel d flags is ignored in this process. The instruction will be executed and a record pulse transmitted even though the output channel d record flag was already set.

Execution time

The timing sequence for this instruction is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. Execution time is constant at five clock periods.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (P) + 1 to P register
Transmit (d) to Q register
Set output channel d record flag
Set (k) to 0

CP03 Transmit record pulse over output channel d cable

CP04 Read storage bank B to X register
Read storage bank B to f and d registers

75XX Pass

76XX Pass

These two instructions are identical and perform no logical function. Each instruction results in a five clock period delay.

Execution time

The timing sequence for these instructions is listed below. The storage banks are designated as bank A and bank B where bank A contains the current instruction word. The storage reference at CP00 must be to bank B for the next instruction word. There can be no delay due to storage bank conflict at this storage reference.

CP00 Transmit (P) + 1 to S register B

CP01 Begin read/write cycle for storage bank B

CP02 Transmit (d) to Q register
Transmit (P) + 1 to P register
Set (k) to 0

CP04 Read storage bank B to X register
Read storage bank B to f and d registers

77XX Error stop

This instruction format causes the PPU program to stop and indicate a program error condition to the MCU. The PPU can be restarted only by a "dead start" condition.

INDEX

INDEX TO INSTRUCTIONS

Central Processor

INDEX TO INSTRUCTIONS

INDEX TO INSTRUCTIONS

Peripheral Processors (Cont'd)