SPERRY RAND

UNIVAC 1108

MULTI-PROCESSOR SYSTEM

PROCESSOR AND STORAGE

PROGRAMMERS REFERENCE

UP-4053 Rev. 1

This manual is published by the Univac Division of Sperry Rand Corporation in loose leaf format. This format provides a rapid and complete means of keeping recipients apprised of UNIVAC® Systems developments. The information presented herein may not reflect the current status of the product. For the current status of the product, contact your local Univac Representative.

The Univac Division will issue updating packages, utilizing primarily a page-for-page or unit replacement technique. Such issuance will provide notification of hardware or software changes and refinements. The Univac Division reserves the right to make such additions, corrections, and/or deletions as, in the judgment of the Univac Division, are required by the development of its Systems.

PRINTED IN U.S.A.

CONTENTS

APPENDIXES

FIGURES

TABLES

1. INTRODUCTION

1.1. GENERAL

This manual provides information on the UNIVAC 1108 Central Processor Unit (CPU), the Multi-Module Access Unit, and the Storage Unit (main storage) of the UNIVAC 1108 Multi-Processor System.

1.2. UNIVAC 1108 MULTI-PROCESSOR SYSTEM

The UNIVAC 1108 Multi-Processor System is the logical successor to the UNIVAC 1107 Thin-Film Memory Computer. It has higher speeds, larger capacities, increased versatility, and smaller physical size than the UNIVAC 1107 System.

Two prime objectives of a multiprocessor system are:

- Increased system performance
- Increased system availability

The UNIVAC 1108 Multi-Processor System achieves increased performance by using multiple processing units, multiple input/output units, and multiple access paths to critical peripheral devices.

Increased availability is achieved by providing multiple access paths to all system components, and by permitting the removal of any system component for offline testing and maintenance while the online units continue to operate.

A UNIVAC 1108 Multi-Processor System includes from one to three Central Processor Units (CPU) and one or two Input/Output Controllers (IOC). In those systems having more than one CPU, the CPU's may operate simultaneously and independently of each other, but they are coordinated by a single Executive program. A system consisting of only one CPU and no IOC is referred to as a Unit Processor System.

Figure 1–1 is a simplified diagram of a UNIVAC 1108 Multi-Processor System. Primary units are four Storage Units, three CPU's, two IOC's, and the Availability Control Unit (ACU). Some of the other possible system configurations are presented and discussed in Section 3.

1.3. SYSTEM COMPONENTS

The components of a UNIVAC 1108 Multi-Processor System are as follows:

- Processor Group — Type No. 3011–95
- Multiprocessor Capability — Feature No. 1053–99
- Storage Group — Type No. 7005–72 (131K)
7005–71 (196K)
7005–70 (262K)
- Multi-Module Access Unit (MMA) — Type No. 0954–99
- I/O Controller (IOC) — Type No. 5013–00
- Availability Control Unit (ACU) — Type No. 2506–00
- Shared Peripheral Interface (SPI) — Type No. 0955–04
- Peripheral Subsystems

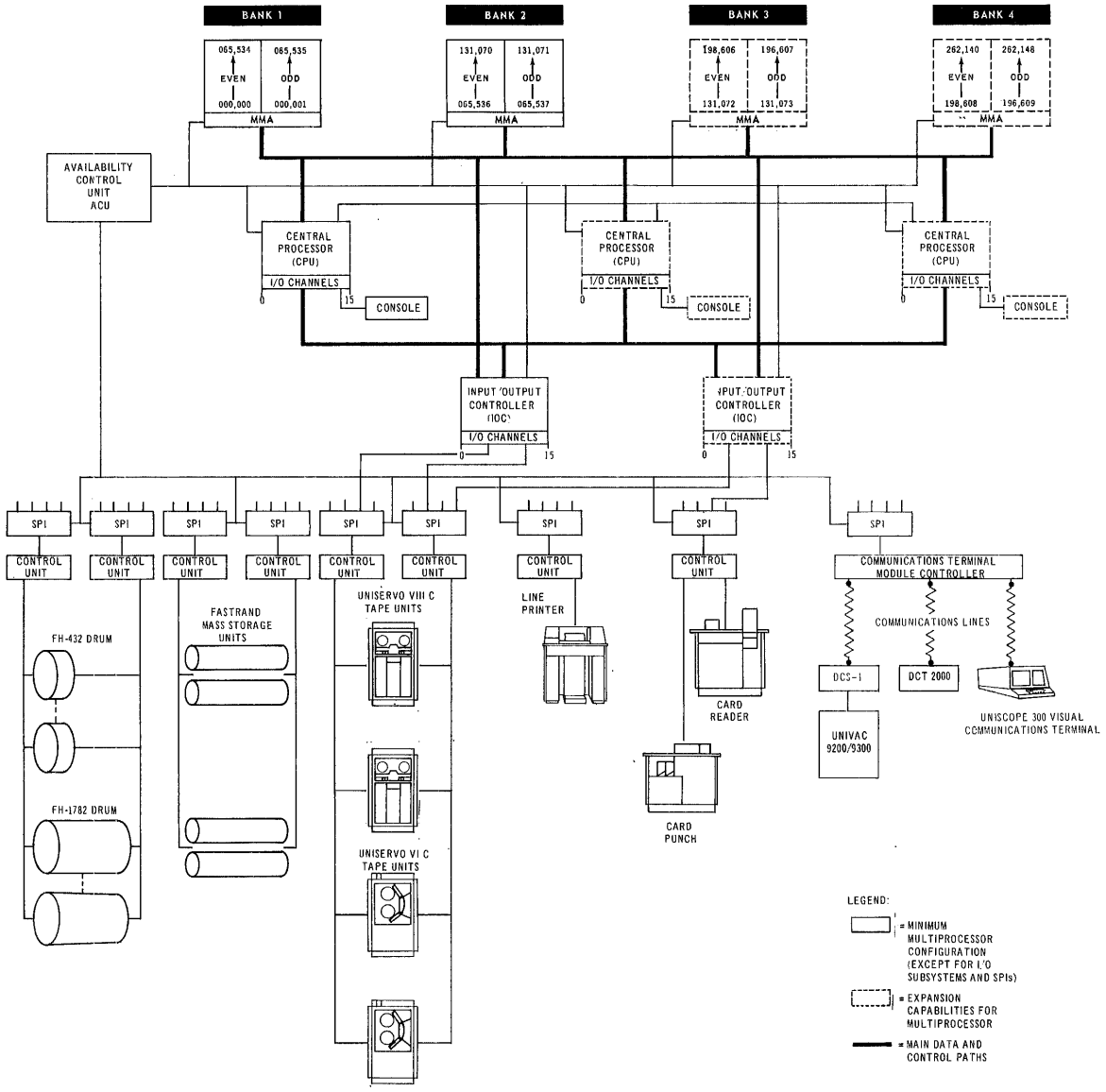


Figure 1–1. UNIVAC 1108 Multi-Processor System

1.3.1. Processor Group

The Processor Group consists of the following (each in a separate cabinet):

- Central Processor Unit (CPU)
- Power and Maintenance Cabinet
- Display Console

The CPU incorporates all features needed for operation in a multiprocessor environment. Each of the CPU's has full access to all of main storage through the Multi-Module Access Units (MMA's). Through its input/output (I/O) section, each CPU has full access to all I/O subsystems in the system.

The Display Console enables the operator to control and communicate with the operating system. Each CPU communicates with its console through I/O channel 15. Output from the CPU to the operator takes the form of messages displayed on a cathode-ray tube (CRT) display, and hardware status information is displayed on the operator's control panel. Operator to CPU communication and control is provided by the console keyboard, and the positioning of various conditioning and directing switches. A hard copy of the messages appearing on the CRT display is provided by a UNIVAC PAGEWRITER printer.

Each Display Console is equipped with a day clock which displays the time of day in hours, minutes, and hundredths of a minute. Storage of the day clock reading at a fixed location in main storage can be enabled or disabled, either manually or under program control. When enabled, the reading is updated 100 times per minute, and interrupts to the CPU are generated at six second intervals.

1.3.2. Storage Group

The Storage Group may consist of from one to four cabinets, each of which houses a Storage Unit and power supply. A Storage Unit provides 65,536 words of main storage (36-bit words). The main storage cycle time is 750 nanoseconds.

By using hardware implemented interleaving for addressing, main storage is utilized more efficiently.

1.3.3. Multi-Module Access Unit (MMA)

The MMA is the interface between a Storage Unit and the CPU's and IOC's in a UNIVAC 1108 Multi-Processor System. Each MMA is housed in a separate cabinet. A MMA includes two logically independent sections, each associated with one module (32K) of a Storage Unit. Each section is used by the CPU's and IOC's to request and receive access to the corresponding module of a Storage Unit. When a conflict occurs between two or more access requests, the MMA services the requests on a priority basis.

1.3.4. Input/Output Controller (IOC)

The IOC is functionally similar to the I/O section of the CPU. The IOC may have up to 16 I/O channels and may be controlled by any of the CPU's in the system. A CPU sends commands to the IOC; this action initiates I/O data transfers under control of the IOC. The data transfers between the subsystems and main storage take place independently of the CPU. When the IOC completes an I/O operation, it signals I/O termination to the CPU. All command and data transfers between the CPU and IOC utilize the CPU's I/O channel interface. The IOC further enhances the system performance by providing chained-buffer (scatter-read/gather-write) operation, and by permitting a more efficient Externally Specified Index (ESI) mode of operation.

One or two IOC's may be included in a UNIVAC 1108 Multi-Processor System. For additional information on the IOC see *UNIVAC 1108 Multi-Processor System Input/Output Controller Programmer/Operator Reference Manual, UP-7514* (current version).

1.3.5. Availability Control Unit (ACU)

The ACU provides complete configuration control of the UNIVAC 1108 Multi-Processor system. It interfaces with the CPU's, IOC's, Storage Units, and MMA's for up to 24 peripheral subsystems. The ACU provides the means for:

- dividing the overall system into independent systems;
- disabling the CPU's or IOC's whenever a power fault occurs;
- taking a unit offline for maintenance without interfering with the operation of the remainder of the system;
- initiating an automatic recovery sequence in the event of system failure; and
- indicating to the CPU which units are online and available.

1.3.6. Shared Peripheral Interface (SPI)

The SPI enables several processors to control a single peripheral subsystem. It is an electronic switching device through which two, three, or four processors (CPU's or IOC's) in any combination can share a peripheral subsystem. Certain peripheral subsystems provide an interface which is functionally equivalent to a SPI.

Additional information on the SPI (formerly called Multiple Processor Adapter) can be found in *UNIVAC 1108 Multi-Processor System Multiple Processor Adapter Programmer/Operator Reference Manual, UP-7562* (current version).

1.3.7. Peripheral Subsystems

Various peripheral subsystems can be used with the UNIVAC 1108 Multi-Processor System. These subsystems include high speed printers, card readers and punches, magnetic tapes, magnetic drums, communication subsystems. In the UNIVAC 1108 Multi-Processor System, the subsystems can be accessed by all the CPU's by utilizing an IOC and/or SPI (or equivalent). A subsystem may also be directly connected to an I/O channel of a CPU. In this case, the subsystem is available only to that CPU.

2. PROCESSOR UNIT

2.1. GENERAL

The Central Processor Unit (CPU) is a solid-state, integrated-circuit, data processing unit which features a powerful and comprehensive instruction repertoire, addressable control registers, high speed arithmetic operations, and a versatile input/output (I/O) section.

2.2. CONTROL SECTION

The control section of the CPU interprets instructions and directs all CPU operations except certain I/O operations. It is discussed briefly below and in more detail in Section 5.

2.2.1. Control Section Operation

The program instruction words are sequentially loaded into the control section. Each instruction word is interpreted by the control section which generates the signals necessary to perform the instruction. The instruction words are located in main storage and the data words (operands) are located either in main storage or in the addressable control registers which are part of the CPU's control section. The control section includes a Program Address Counter (P register), which addresses main storage to obtain the instruction words.

The instruction word is divided into fields. These fields specify to the control section the function to be performed, which portion of the operand is to be used, a control register, indexing, index register modification, indirect addressing, and an operand address.

2.2.2. Instruction Repertoire

The instruction repertoire is all the instructions that the control section of the CPU is capable of interpreting and executing. It includes fixed-point and floating-point arithmetic, logical functions, data transfers, block transfers, comparisons, tests, I/O control, and special purpose instructions. There are over 140 basic instructions in the repertoire. Provision has been made for partial word data transfers and for repetitive operations. Indexing capability is provided with all instructions. Indirect addressing capability is also provided and is usable to any level with full indexing capability at each level.

Instructions such as data transfers, single-precision fixed-point adds, certain logical functions, require only 750 nanoseconds for complete execution. Indexing does not add to the execution time of an instruction.

2.2.3. Control Registers

The 128 addressable control registers of the control section are integrated-circuit registers. These control registers are addressed either explicitly or implicitly by the instructions. They fall into five categories: index registers, arithmetic registers, special registers, I/O access control registers, and unassigned registers.

The control registers are discussed in detail in 3.3 and Section 5.

2.3. ARITHMETIC SECTION

All arithmetic computation is performed using the nonaddressable registers of the arithmetic section. These arithmetic processes can be performed in either fixed-point or floating-point mode. Fixed-point arithmetic instructions provide for single-precision, double-precision, half-word, and third-word addition and subtraction, and for fraction and integer multiplication and division. Floating-point instructions provide for both single-precision and double-precision operation. The arithmetic section also performs certain logical operations such as shifting and comparisons. The instruction word may be used to specify the transfer of any chosen portion of a word (half, third, quarter, or sixth) to or from the arithmetic section. The ability to transfer only the selected portion of a word minimizes the number of masking and shifting operations required.

A shift matrix in the arithmetic section permits the completion of an entire single word shift operation in one main storage cycle time. By use of the matrix, the shift operation can shift a single or double word operand in either direction up to 72 bit positions. Partial word transfers utilize automatic shifting operations in the shift matrix.

Details on the operation of the arithmetic section of the CPU are found in Section 4.

2.4. INPUT/OUTPUT SECTION

The I/O section, while physically located in the CPU cabinet, is a functional entity. I/O activity is initiated when the interpretation of certain instructions by the control section causes signals to be sent to the I/O section. Once the I/O operation is initiated, the I/O section and the subsystem control the input and output transfers. The I/O section complements the control and arithmetic sections' orientation toward high speed concurrent program processing and real time operation. The I/O section operates with a wide variety of peripheral devices, and it requires minimal attention from the control section.

Once an I/O operation is initiated by the program, I/O activity is independent of program control and it is controlled by an I/O Access Control Word stored in a control register. The I/O data flows between the main storage and the peripheral subsystem through an I/O channel. Each I/O channel consists of 36 input data lines, 36 output data lines, and various control signal lines. All data word bits are transmitted in parallel to or from the subsystem, but data can flow in only one direction over a channel at any given instant.

The I/O section of a basic CPU includes eight I/O channels. An I/O Channel Expansion Feature (F0680) may be added to the CPU to increase the number of I/O channels from eight to 12. If 16 I/O channels are required, a second I/O Channel Expansion Feature must be added. The total number of I/O channels available to a CPU is expanded by the inclusion of one or two IOC's in the system.

Details on the input/output section of the CPU are presented in Section 7.

3. STORAGE

3.1. GENERAL

The Storage Units of the UNIVAC 1108 Multi-Processor System provide the main storage for instruction and data words. The 128 addressable control registers in the control section of each Central Processor Unit (CPU) provide fast access storage for data and control words.

3.2. MAIN STORAGE

A main storage word consists of 36 information bits and 2 parity bits. The two parity bits provide independent hardware parity checking of the leftmost 18 data bits and the rightmost 18 data bits of the main storage word, respectively. The word may be an instruction word, a data word, or an access control word.

A Storage Unit can store 65,536 words. Functionally, each Storage Unit is divided into two modules, with each module providing 32,768 words of main storage. Each module includes independent accessing circuitry which permits simultaneous accessing of multiple modules.

3.2.1. Addressing Main Storage

Main storage addressing may be either interleaved or noninterleaved. The choice is made by implementing a modification in the CPU.

3.2.1.1. Noninterleaved Addressing of Main Storage

Noninterleaved addressing is provided only for a UNIVAC 1108 Unit Processor System (defined as a UNIVAC 1108 System which consists of one CPU and no IOC's) which has 65,536 or 131,072 words of main storage. In noninterleaved addressing, the main storage modules and the locations within each module are addressed sequentially. A Storage Unit cabinet containing two 32K modules is the minimum complement of main storage available. The maximum complement of 131K words of noninterleaved main storage is obtained by the addition of another cabinet containing 65K words of main storage. Table 3–1 shows the range of addresses for the locations in each main storage module.

LOGICAL MODULE NUMBER	ADDRESS RANGES	
	OCTAL	DECIMAL
0	000000–077777	0–32,767
1	100000–177777	32,768–65,535
2	200000–277777	65,536–98,303
3	300000–377777	98,304–131,071

Table 3–1. Octal and Decimal Address Ranges for Noninterleaved Main Storage Modules

A CPU can retrieve or store the operand of the current instruction, while simultaneously obtaining the next instruction from main storage (overlapping). If the operand for the current instruction and the next instruction word are in different modules, the CPU simultaneously requests main storage access for both the operand and the next instruction word (alternate module operation). If the operand for the current instruction and the next instruction word are in the same module of main storage, simultaneous access is not possible, and the CPU requests main storage access (read or write) for the operand first. After the operand has been transferred, the CPU requests main storage access to retrieve the next instruction word.

Figure 3–1 is a block diagram of a CPU with the maximum complement of noninterleaved main storage. Since the desired efficiency of alternate module operation is obtained only when the instruction words and operands (data words) are not in the same main storage module, Main Storage Modules #0 and #1 in Figure 3–1 provide the minimum storage requirement for alternate module operation. Modules #2 and #3 provide expansion for the noninterleaved addressed main storage.

3.2.1.2. Interleaved Addressing of Main Storage

The interleaved addressing method permits a more efficient use of main storage in the multiprocessor environment. Interleaved addressing is always used with a multiprocessor system and also with a unit processor system which has 196,608 or 262,144 words of main storage. When interleaved addressing is used, each processor (CPU or IOC) can address up to 262,144 main storage words.

When interleaved addressing is used, sequentially addressed main storage words within a 65K Storage Unit are located alternately in the two modules, even number addresses being in one 32K module and odd number addresses in the other.

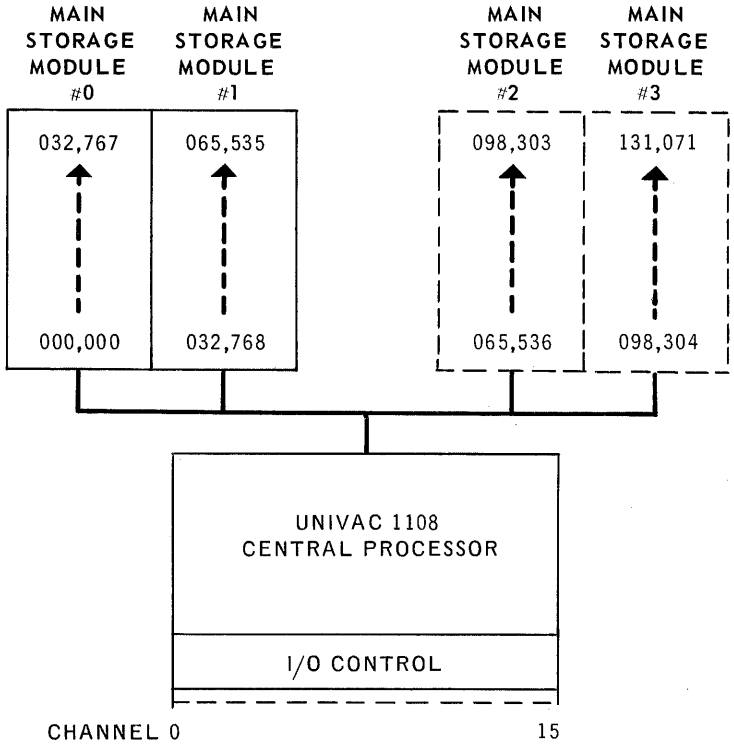


Figure 3–1. Central Processor Unit with Maximum Noninterleaved Main Storage

At least two Storage Units (131,072 words) are necessary for interleaved addressing. Three or four Storage Units may be used, thereby expanding the capacity to 196K or 262K words. Table 3–2 shows the range of addresses for each module pair. Figure 3–2 shows a CPU with the maximum complement of interleaved addressed main storage.

LOGICAL MODULE PAIR NUMBER	ADDRESS RANGES	
	OCTAL	DECIMAL
0	000000–177777	0–65,535
1	200000–377777	65,536–131,071
2	400000–577777	131,072–196,607
3	600000–777777	196,608–262,143

Table 3–2. Octal and Decimal Address Ranges for Interleaved Main Storage Modules

Figure 3–3 shows a CPU and IOC of a UNIVAC 1108 Multi-Processor System with the maximum complement of interleaved main storage. The IOC requests access to main storage independently of the CPU. The IOC always provides for interleaved addressing of main storage.

Figure 1–1 shows a UNIVAC 1108 Multi-Processor System consisting of the maximum complement of interleaved addressed main storage with two IOC's and three CPU's. Each IOC and CPU has access to each main storage module by way of the MMA's.

The advantage of interleaved addressing is that it permits simultaneous access by two processors (CPU or IOC) to any odd address and any even address in a main storage module pair. The simultaneity of main storage accessing provides service under ideal conditions to a pair of access-requesting units at a rate which meets their needs without a perceptible decrease in overall operating speed for either unit.

3.2.2. Multi-Module Access Unit (MMA)

The MMA is the interface between the two main storage modules of a 65K Storage Unit and a combination of IOC's and CPU's. The MMA includes switching and priority networks that provide access to each of two main storage modules for as many as two IOC's and three CPU's (IOC #0, IOC #1, CPU #0, CPU #1, and CPU #2).

The addition of the MMA Expansion Feature (F0879) expands the interface for that MMA to accommodate an additional CPU and IOC.

The MMA has two sections, one for each 32K main storage module of a Storage Unit. When an MMA receives an access request from an IOC or CPU for a main storage module which is not active, it initiates a cycle for that module. The MMA resolves conflicts caused by receipt of one or more additional access requests for that module within approximately 15 nanoseconds of the initial request.

When the MMA initiates a main storage cycle, the priority circuits associated with that module are activated to handle any access requests pending for that module at the time the cycle is initiated or any access requests received for that module during the current cycle.

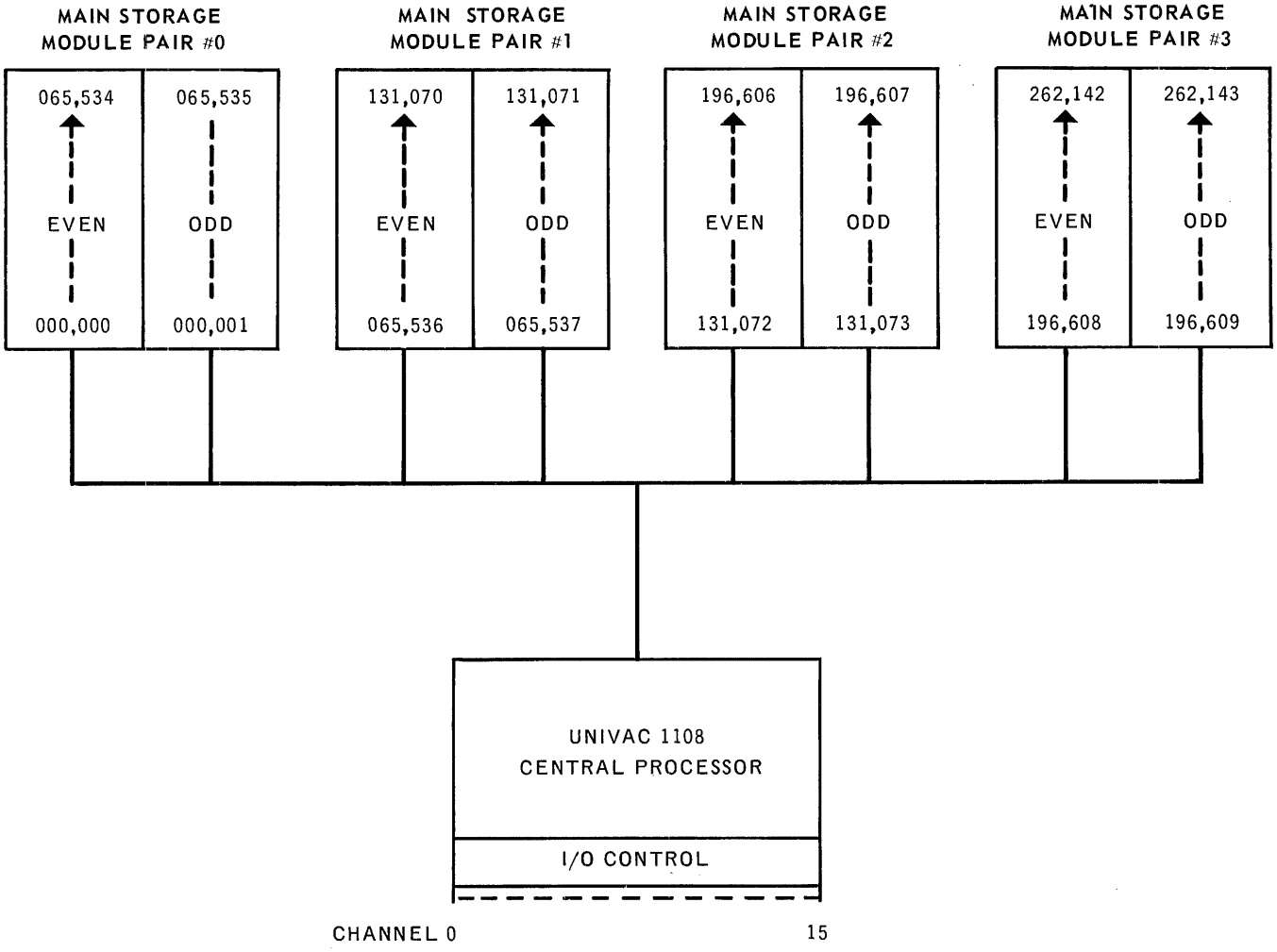


Figure 3–2. CPU with Maximum Interleaved Main Storage

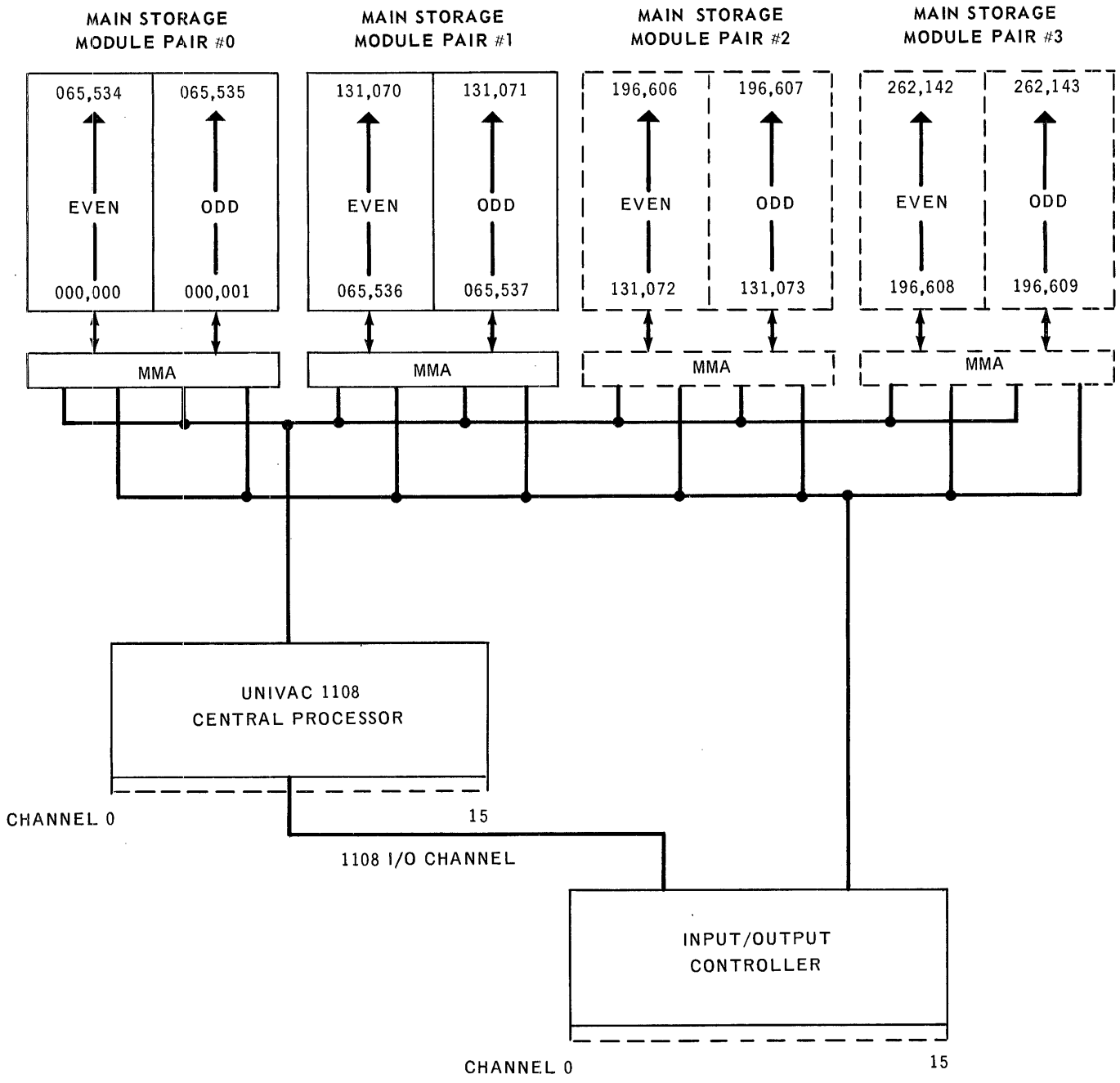


Figure 3–3. CPU, IOC, and Main Storage with MMA's for UNIVAC 1108 Multi-Processor System

If only one access request is pending when a cycle in a particular module is completed, the pending access request is serviced next. If access requests from two or more processors (CPU's or IOC's) are pending for a module just prior to completing a cycle in that module, the MMA's priority network determines which processor is serviced next, as follows:

- IOC's always have higher priority than CPU's.
- IOC #0 has higher priority than IOC #1.
- CPU #0 has higher priority than CPU #1 which in turn has higher priority than CPU #2.

Summarizing, the basic priority sequence for access to a main storage module is: IOC #0, IOC #1, CPU #0, CPU #1, and CPU #2 (highest to lowest priority).

Deviations from this basic priority scheme are as follows:

- If both IOC's are repetitively requesting access to a main storage module, priority alternates from IOC #0 to IOC #1 to IOC #0, etc; IOC #0 cannot monopolize the use of a main storage module to the exclusion of IOC #1. This occurs because the MMA priority circuits determine which IOC is to be granted access during the current main storage access cycle, that is, before the IOC currently being serviced can present another access request. If the two IOC's repetitively request access to the same module, however, access to that module by any of the CPU's is delayed until there is a break in the series of IOC access requests.
- If one or more CPU's request access to a main storage module while that module is servicing an IOC, and the other IOC is not requesting access, the highest priority CPU (considering lockout discussed below) will gain access next. The MMA priority circuits determine which processor will be serviced next before the completion of the current servicing cycle, that is, before the active IOC can present another access request for that module.

It is not possible for a CPU to monopolize the use of a main storage module to the exclusion of a lower priority CPU. The priority network modifies the basic priority scheme to prevent this by setting and clearing lockouts, as follows:

- Set Lockout for CPU #0: During each cycle servicing an access request from CPU #0, a lockout is set to inhibit initiation of a subsequent cycle for CPU #0 in the main storage module if either CPU #1 or CPU #2 has a pending access request for that module.
- Clear Lockout for CPU #0: If the lockout is set for CPU #0, it is cleared during any subsequent cycle in the module when CPU #1 is serviced provided there is no pending access request for that module from CPU #2. It is also cleared during any subsequent cycle in the module servicing CPU #2.
- Set Lockout for CPU #1: During each cycle servicing an access request from CPU #1, a lockout is set to inhibit initiation of a subsequent cycle for CPU #1 in the module if and only if CPU #2 has a pending access request for that module.
- Clear Lockout for CPU #1: If the lockout is set for CPU #1, it is cleared during any subsequent cycle when that module is used to service CPU #2.

3.2.3. Main Storage Special Address Assignments

There are two special categories of main storage addresses. These are main storage addresses 0_8 through 177_8, and a set of main storage addresses with fixed assignments.

3.2.3.1. Hidden Storage

The control registers are assigned the same addresses as main storage locations 0_8 through 177_8. If data is written into or read from a location whose operand address is less than 200_8, the data is transferred to or from that addressed location in the control registers rather than to or from a location in main storage except as noted in the following paragraph. Thus, locations 0_8 through 177_8 in main storage are somewhat protected and are referred to as *hidden storage*.

Information is read from locations in hidden storage rather than from the control registers in the following instances:

- The address is that of an instruction defined by the contents of the Program Address Counter (P register).
- The address is that of a word to be referenced as part of an indirect addressing sequence.
- The address is that of an instruction word referenced by an Execute instruction.

Information can be read from or written into hidden storage through data transfers performed by an I/O operation. (An I/O operation never transfers data to or from the control registers.)

3.2.3.2. Fixed Address Assignments

The interrupt subroutine entrances and status words are assigned fixed locations in main storage as shown in Table 3–3. The values for bits 15, 16, and 17 of the addresses listed in Table 3–3 are represented by a X, Y, or Z in the address.

The value X is the contents of the Memory Select Register (MSR), which may be manually loaded by depressing the desired combination of the three MSR switches on the maintenance panel. It can also be loaded under program control by performing the Select Interrupt Location instruction. In a system with noninterleaved addressing and the maximum complement of 131,072 words of main storage, the only values permitted in the MSR are 0, 1, 2, or 3 (see Table 3–4). In a system with interleaved addressing and the maximum complement of 262,144 words of main storage, the only values permitted in the MSR are 0, 2, 4, or 6 (see Table 3–5). When an initial load operation is performed, the value in the MSR identifies the main storage module or module pair in which the incoming data is to be stored. During an ESI-I/O operation using the CPU's I/O section, the value in the MSR identifies the main storage module or module pair from which the ESI access control words are obtained.

The value Y is determined from the contents of the Last Address Register (LAR) ORed with the settings of the leftmost three Last Address switches on the CPU's maintenance panel as explained in 8.3.1.3. The value A is 3 or 7 depending on the setting of the rightmost Last Address switch on the CPU's maintenance panel.

ADDRESS (OCTAL)	ASSIGNMENT
X00200	CPU #0 External Interrupt Status Word
X00201	CPU #1 External Interrupt Status Word
X00202	CPU #2 External Interrupt Status Word
X00203	Unassigned
X00204	Unassigned
X00205	Unassigned
X00206	Unassigned
X00207	Unassigned
X00210	Power Loss Interrupt
X00211	ESI Access Control Word Parity Error Interrupt
X00212	ISI Access Control Word Parity Error Interrupt
X00213	I/O Data Parity Error Interrupt
X00214	Unassigned
X00215	Unassigned
X00216	Day Clock Input
X00217	Day Clock Interrupt
X00220	ISI Input Monitor Interrupt
X00221	ISI Output Monitor Interrupt
X00222	ISI Function Monitor Interrupt
X00223	ISI External Interrupt
X00224	ESI Input Monitor Interrupt
X00225	ESI Output Monitor Interrupt
X00226	Unassigned
X00227	ESI External Interrupt
X00230	External Interrupt Status Word for CPU Type 3011–99
X00231	Real Time Clock Interrupt
X00232	Interprocessor Interrupt #0
X00233	Interprocessor Interrupt #1
X00234	Unassigned
X00235	Main Storage Parity Error Interrupt, Mem 2
X00236	Main Storage Parity Error Interrupt, Mem 3 (see Tables 3–4 and 3–5)
X00237	Main Storage Parity Error Interrupt, Mem 4
X00240	Control Register Parity Error Interrupt
X00241	Illegal Instruction Fault Interrupt
X00242	Executive Return Interrupt
X00243	Guard Mode/Storage Limits Protection Fault Interrupt
X00244	Test and Set Interrupt
X00245	Floating-Point Characteristic Underflow Fault Interrupt
X00246	Floating-Point Characteristic Overflow Fault Interrupt
X00247	Divide Fault Interrupt
X00250	Unassigned
X00251	Unassigned
X00252	Unassigned
X00253	Unassigned
X00254	Unassigned
X00255	Unassigned
X00256	Unassigned
X00257	Unassigned
Z00260 – Z00377	IOC External Interrupt Status Word Locations
YA7776	Main Storage Parity Error Interrupt, Mem 1 (see Tables 3–4 and 3–5)

Table 3–3. Main Storage Fixed Address Assignments

The value Z is supplied by the contents of the Interrupt Bias Register in the IOC. See *UNIVAC 1108 Multi-Processor System Input/Output Controller Programmer/Operator Reference Manual, UP-7514* (current version).

MSR VALUE	LOGICAL MODULE #	ADDRESS RANGES (OCTAL)	PHYSICAL MODULE #
0	0	0–077777	MEM 1
1	1	100000–177777	MEM 2
2	2	200000–277777	MEM 3
3	3	300000–377777	MEM 4

Table 3–4. MSR Values vs. Module Pair Identification for Noninterleaved Main Storage

MSR VALUE	LOGICAL MODULE PAIR #	ADDRESS RANGES (OCTAL)	PHYSICAL MODULE PAIR #
0	0	0–177777	MEM 1
2	1	200000–377777	MEM 3
4	2	400000–577777	MEM 2
6	3	600000–777777	MEM 4

Table 3–5. MSR Values vs. Module Pair Identification for Interleaved Main Storage

3.2.4. Main Storage Protection

Main storage protection is provided to prevent an operating program from writing into, reading from, or jumping into an unrelated portion of main storage. Main storage protection can be established in either of two modes. One mode affords write, read, and jump protection; the other affords protection only when writing.

Main storage protection is controlled through a combination of the Storage Limits Register (SLR) and the guard mode or write only storage protection designators of the Processor State Register (PSR). The address ranges of the protected portions of main storage are specified by storing in the SLR the upper and lower absolute address limits of the two portions of main storage in which the instructions and data for the currently operating program are located. Granularity of the designated areas is 512 words. The two portions can be completely overlapping, partially overlapping, or distinctly separate. The mode of storage protection and whether or not it is enforced depends on the condition of the guard mode and the write only storage protection designators (D2 and D3) stored in the PSR.

3.3. CONTROL REGISTERS

The control section of the CPU includes 128 addressable control registers. Each control register stores a word consisting of 36 information bits and 2 parity bits. The two parity bits provide independent hardware parity checking of the leftmost 18 information bits and the rightmost 18 information bits of the word in the control register. The control registers are addressable by the ja, a, and x fields of the instruction word and by the value U developed in the index subsection of the CPU's control section. The details of control register addressing are explained in Section 5. Table 3–6 summarizes the control register address assignments.

3.3.1. Control Register Selection Designator

The 128 addressable control registers include one set of registers for use by the user program and another set for use by the Executive program. The control register selection designator (D6) in the PSR defines which set of registers is addressed by the a and x designators of an instruction word. When D6 = 0, the user program set of control registers is addressed; when D6 = 1, the Executive program set of control registers is addressed. The contents of D6 has no effect on the choice of a control register for any particular ja combination or value of U.

NO. OF REGISTERS	ADDRESS (OCTAL)	ASSIGNMENT
1	000	PSR Temporary Storage Register
15	001–017	User X Registers
16	014–033	User A Registers
4	034–037	User Unassigned Registers
16	040–057	Input Access Control Word Registers
16	060–077	Output Access Control Word Registers
1	100	R0 – Real Time Clock
1	101	User R1 – Repeat Count Register
1	102	User R2 – Mask Register
13	103–117	User Unassigned R registers
1	120	Executive R Register
1	121	Executive R1 – Repeat Count Register
1	122	Executive R2 – Mask Register
13	123–137	Executive R Registers
1	140	Executive Nonindexing X Register
15	141–157	Executive X Registers
16	154–173	Executive A Registers
4	174–177	Executive Unassigned Registers

Table 3–6. Control Register Address Assignments

3.3.2. Control Register Address Assignments

Operand address 0_8 through 177_8 are assigned to the control registers. The following paragraphs define the various uses and related address assignments for the control registers.

3.3.2.1. Temporary Storage for Processor State Register (PSR) at Interrupt – Address 000_8

When an interrupt occurs, the contents of the PSR are automatically stored in this control register. Obviously, this register must not be used to store program information because such information would be destroyed whenever an interrupt occurs.

3.3.2.2. User Index (X) Registers – Addresses 001_8–017_8

The X registers normally contain the modifiers used in indexing operations. The index register stores an 18-bit modifier (Xm in bits 17–0), and an 18-bit increment (Xi in bits 35–18).

3.3.2.3. User Accumulator (A) Registers – Addresses 014_8–033_8

The A registers store arithmetic operands and results. The actual computation or logical function is performed in the arithmetic section and the results are stored in the A register or registers specified by the instruction. Four of the A registers (addresses 014_8–017_8) overlap registers assigned as X registers. This affords additional versatility in the use of A registers and X registers.

3.3.2.4. User Unassigned Registers – Addresses 034_8–037_8

Two of these unassigned registers (034_8 and 035_8) serve as an extension of the set of user A registers when D6 = 0 and an instruction which requires more than one user A register is being performed. All four of these unassigned registers can serve as general purpose registers.

3.3.2.5. Input/Output Access Control Registers – Addresses 040_8–077_8

The contents of the I/O Access Control Registers are used to control the word-by-word data transmissions over the CPU I/O channels. Two of these registers are assigned to each of the 16 CPU I/O channels as follows:

- The registers at addresses 040_8 through 057_8 are Input Access Control Registers for I/O channels 0 through 15.
- The registers at addresses 060_8 through 077_8 are Output Access Control Registers for I/O channels 0 through 15.

For Internally Specified Index (ISI) I/O operations, the contents of the Input Access Control Registers are used to control the addressing of main storage locations when input data words are received from the subsystems (through the CPU's I/O section). The contents of the Output Access Control Registers are used to control the addressing of main storage locations from which output data words and function words are sent to the subsystems through the CPU's I/O section (see 7.2).

For Externally Specified Index (ESI) I/O operations, the Input Access Control Register for each I/O channel operating in the ESI mode is used to identify the address of the most recently used Input Access Control Word and the most recently used Output Access Control Word for that I/O channel. The contents of the Output Access Control Word for each I/O channel operating in the ESI mode are used to address a function word in main storage which is to be sent to the subsystem connected to the CPU through the I/O channel (see 7.3).

3.3.2.6. Real Time Clock (R0) Register – Address 100_8

The contents of the lower half (bits 17–0) of the Real Time Clock (RTC) register are decremented by one every 200 microseconds, independently of program control or supervision. The CPU time utilized for each decrementation cycle is 0.375 microseconds. A real time clock interrupt occurs if the RTC value in the lower half of the RTC register is zero when a decrementation cycle is initiated. It is recommended that the programmer avoid using the upper half (bits 35–18) of the RTC register as a data storage area. Whenever the contents of the RTC register are read out, the contents of the upper half of the register should be ignored.

3.3.2.7. User Repeat Count (R1) Register – Address 101_8

The contents of the lower half (bits 17–0) of the Repeat Count Register (RCR) are called the repeat count. When a repeat instruction is performed, the repeat count is decremented by one for each pass through the instruction. If an interrupt occurs during the iterations, the repeat sequence is suspended to process the interrupt. The current repeat count is available in R1. The repeat sequence is normally resumed after the interrupt has been processed.

If the repeat count is +0 when a repeated instruction is first initiated, the instruction is not performed, and the next instruction is initiated. When the repeat count is decremented to +0 during a pass through a repeated instruction, the pass is completed, but another iteration is not initiated. If a pass through a repeated instruction leads to termination of the iterations before the repeat count has been decremented to +0, the repeat count after decrementation for that pass is available in the RCR. It is recommended that the programmer avoid the use of the upper half (bits 35–18) of the RCR for storing data, and the contents of the upper half should be ignored whenever the register is read.

3.3.2.8. User Mask (R2) Register – Address 102_8

The bits in the Mask Register specify the bits of the operand to be operated upon in certain instructions. A logical AND is performed with the operand and the mask or its complement. The selected portions of the operands are then used when the instruction is executed.

3.3.2.9. User R Registers – Addresses 103_8–117_8

These registers are unassigned and serve as general.purpose registers. When D6 (PSR) = 0, each of these registers can be implicitly addressed by one of the values 3_8 through 17_8 in the a field of a Load R or Store R instruction.

3.3.2.10. Executive R Register – Address 120_8

This register is unassigned and serves as a general purpose register. When D6 (PSR) = 1, this register is implicitly addressed when the a field of a Load R or Store R instruction equals zero.

3.3.2.11. Executive Repeat Count (R1) Register – Address 121_8

This register has the same function and format as the User R1 – Repeat Count Register (see 3.3.2.7).

3.3.2.12. Executive Mask (R2) Register – Address 122_8

This register performs the same function as the User R2 – Mask Register (see 3.3.2.8).

3.3.2.13. Executive R Registers – Addresses 123_8–137_8

These registers are unassigned and serve as general purpose registers. When D6 (PSR) = 1, each of these registers can be implicitly addressed by one of the values 3_8 through 17_8 in the a field of a Load R or Store R instruction.

3.3.2.14. Executive Nonindexing X Register – Address 140_8

This register is assigned as an X register. It provides an indexing capability only if a = 0 and x $\neq$ 0 in a Block Transfer instruction when D6 = 1. It does not provide an indexing capability when addressed by the x field of an instruction word. It may also be used as a general purpose register.

3.3.2.15. Executive Index (X) Registers – Addresses 141_8–157_8

These registers perform the same function as the user Index Registers (see 3.3.2.2).

3.3.2.16. Executive Accumulator (A) Registers – Addresses 154_8–173_8

These registers perform the same function as the user A registers (see 3.3.2.3).

3.3.2.17. Executive Unassigned Registers – Addresses 174_8--177_8

These registers are used in the same manner as the unassigned registers at addresses 034–037_8 (see 3.3.2.4).

3.3.3. Control Register Protection

When operating in the guard mode (D2 of the PSR = 1 and D6 = 0; see 8.3.2.2), a guard mode fault interrupt occurs when an attempt is made to store data into any one of the Input Access Control Word Registers (addresses 040_8–057_8), Output Access Control Word Registers (addresses 060_8–077_8), Real Time Clock Register (address 100_8), or any of the Executive registers (addresses 120_8–177_8).

4. CPU ARITHMETIC SECTION

4.1. GENERAL OPERATION

During the execution of most instructions, either one or two operands are transferred from addressable locations to nonaddressable registers in the arithmetic section of the 1108 Central Processor Unit (CPU). When the specified operation is completed, the resultant word or words are stored in addressable storage locations. This section describes the operation of the arithmetic section during the execution of Add, Add Negative (subtract), Multiply and Divide instructions.

4.2. MAIN ADDER

The main adder, in the arithmetic section, is used to perform the Add, Add Negative, Multiply, and Divide instructions. It operates with either 36-bit or 72-bit operands. It performs ones complement subtractive binary arithmetic in the parallel mode. Its processes are always basically subtractive. For example:

- Subtract (add negative) operations are performed in a straightforward manner using the minuend and subtrahend as main adder inputs.
- Add operations are performed by treating the augend as the main adder minuend input and the ones complement of the addend as the main adder subtrahend input.

While the hardware is basically subtractive, the results of an add or subtract operation can be determined by the following rules:

- Addition – add the augend to the addend.
- Subtraction – form the ones complement of the subtrahend and add it to the minuend.
- Exception – if a number is added to itself, the result is all zeros (operation is equivalent to subtracting the number from itself).

These rules represent in reality ones complement additive arithmetic as is illustrated in the following examples.

Example 1 (Addition):

Augend	010 000
Addend	000 111
Sum	010 111

Example 2 (Addition):

Augend	111 110
Addend	111 110
End-around carry	111 100
	⌐→ 1
Sum	111 101

Example 3 (Subtraction):

Minuend	101 011	Minuend	101 011
Subtrahend	110 011	Complement subtrahend and add	001 100
		Difference	110 111

Example 4 (Subtraction):

Minuend	111 110	Minuend	111 110
Subtrahend	111 001	Complement subtrahend and add	000 110
		End-around carry	⌐ 000 100
			⌐→ 1
		Difference	000 101

Example 5 (Addition):

Augend	111 000
Addend	000 111
Sum	000 000

In Example 5, the expected result is a sum consisting of all 1 bits; however, this example points out the exception in which the result produced by a ones complement subtractive adder differs from that produced by ones complement additive arithmetic. This exception to the rules can be stated as follows: if a number is added to its complement, the resultant number is all 0 bits.

The ones complement subtractive process performed in the main adder must be considered in two phases for complete understanding. In addition, the minuend input is the augend, and ones complement of the addend is the subtrahend input. In the first phase, an exclusive OR of the minuend and subtrahend is performed and borrows are generated. For each bit, the first phase result is as follows:

- The result is a 0 bit if the main adder inputs for that bit are identical.
- The result is a 1 bit if the main adder inputs for that bit are not identical.
- A borrow is generated if the minuend input is a 0 bit and the subtrahend input is a 1 bit. This is indicated by the b below certain bits in the examples that follow.

The second phase of the process consists of handling the borrows generated during the first phase. When a borrow is generated, each bit to the left of that bit position is complemented in sequence until a 1 bit is changed to a 0 bit, which halts the action of that borrow. If the leftmost bit of the first phase result is a 0 bit and it is complemented as a result of sequential borrow action, the borrow propagates around the end and continues at the rightmost bit position. If a borrow is generated for the leftmost bit position of the first phase result, the action of that borrow starts at the rightmost bit position.

No special treatment is given to the leftmost bit by the main adder in obtaining the first result. The only special treatment given to the leftmost bit in obtaining the second phase result is the end-around borrow action.

Example 1 (Addition):

Augend	010 000	Augend becomes minuend	010 000
Addend	000 111	Complemented addend becomes subtrahend	111 000
		First phase: minuend XOR subtrahend	101 000
		Borrows	b b
		Second phase results	010 111

Example 2 (Addition):

Augend	111 110	Augend becomes minuend	111 110
Addend	111 110	Complemented addend becomes subtrahend	000 001
		First phase: minuend XOR subtrahend	111 111
		Borrows	b
		Second phase results	111 101

Example 3 (Subtraction):

Minuend		101 011
Subtrahend		110 011
First phase: minuend XOR subtrahend		011 000
Borrows		b
Second phase result		110 111

Example 4 (Subtraction):

Minuend		111 110
Subtrahend		111 001
First phase: minuend XOR subtrahend		000 111
Borrow		b
Second phase result		000 101

Example 5 (Addition):

Augend	111 000	Augend becomes minuend	111 000
Addend	000 111	Complemented addend becomes subtrahend	111 000
		First phase minuend XOR subtrahend	000 000
		No borrows	
		Second phase result	000 000

Example 5 illustrates how the all-zero result is obtained when a number is added to its complement in the ones complement subtractive process. In all other instances, the results produced by the ones complement subtractive process are identical to those produced by the ones complement additive process.

4.2.1. Signed Numbers

While the main adder does not recognize the leftmost bit as being the sign bit, it is always considered to be the sign associated with an operand when that operand is used to represent a number. A 0 bit in this position is considered to be a + (plus) sign indicating a positive number; a 1 bit is considered to be a - (minus) sign indicating a negative number. A 36-bit word consisting of the bits 00.....0101 represents the signed number $+5_{10}$. The ones complement of this word is 11.....1010 and it represents the signed number -5_{10}. The most significant bit of the signed number is defined as the first bit to the right of the sign bit which differs from the sign bit. All bits to the left of the most significant bit are sign bits. In the examples of the signed numbers (+5 and -5), the most significant bit (MSB) is as follows:

```
00....0101          11....11010
   ↑                    ↑
  MSB                  MSB
```

The main adder produces the expected natural signed result when the program considers the operands to be signed numbers except for those cases in which the signed result cannot be adequately expressed in the number of bit positions provided. These are the cases in which an unnatural sign bit appears in the result. If the program maintains full cognizance of the true implied sign, however, such cases are no serious handicap. For example, if the augend for an Add instruction is the largest possible positive single-precision signed number and the addend is the number +1, the resultant sum would be as follows:

Augend	011 111 111 111 111 111 111 111 111 111 111 111
Addend	000 000 000 000 000 000 000 000 000 000 000 001
Sum	100 000 000 000 000 000 000 000 000 000 000 000

In this example, the sum has an unnatural sign because a significant data bit has overflowed into the sign position. The programmer may detect this condition by testing the overflow designator. This designator is set whenever overflow occurs; however, the result as it stands can be considered the numerical sum if the programmer recognizes that the leftmost bit is part of the numerical value and if he is maintaining the sign of his result separately from the numerical part.

4.2.2. Zero as a Signed Number

A signed number which consists of all 0 bits represents the number +0. The ones complement of this word is a word of all 1 bits and it represents the signed number -0.

Zero operands follow the main adder addition rules given in 4.2. The results obtained when operating with two zero operands are as follows:

(+0) + (+0) = +0	(-0) + (+0) = +0
(+0) - (-0) = +0	(-0) - (-0) = +0
(+0) + (-0) = +0	(-0) + (-0) = -0
(+0) - (+0) = +0	(-0) - (+0) = -0

4.3. FIXED-POINT ARITHMETIC

The arithmetic section provides the needed hardware to perform add, add negative (subtract), multiply, and divide operations. All of these operations utilize the main adder. The basic operation performed by the main adder is always subtraction. The detailed explanation of the various operations presented below frequently state that "the contents of one register are added to the contents of another register". The operation is presented in terms of addition for better understanding of the logic flow. The actual operation performed by the main adder is the subtraction of the ones complement of the contents of one register from the contents of another register.

4.3.1. Single-Precision Fixed-Point Addition and Subtraction

The steps performed in the arithmetic section for single-precision Add and Add Negative instructions are as follows:

(1) The 36-bit augend/minuend is transferred from a control register to the augend/minuend register in the arithmetic section.

(2) The addend/subtrahend is transferred from main storage, a control register, or the index subsection to the addend/subtrahend register in the arithmetic section.

- If the addend/subtrahend is a partial word, it is extended to become a full 36-bit word as specified by the j field of the instruction. (See 5.2.2.1.)
- If the instruction is Add Magnitude or Add Negative Magnitude To A and the leftmost bit of the full word addend/subtrahend is a 1 bit, the contents of the addend/subtrahend register are complemented.

(3) The contents of the addend/subtrahend register are added to or subtracted from the contents of the augend/minuend register. The two phases of operation of the main adder proceed as explained in 4.2 with the following exceptions:

- If the instruction is Add Halves or Add Negative Halves, a borrow generated for or through bit 17 is propagated to bit 0 rather than to bit 18. A borrow generated for or through bit 35 is propagated to bit 18 rather than to bit 0.
- If the instruction is Add Thirds or Add Negative Thirds, a borrow generated for or through bits 11, 23, or 35 propagates to bits 0, 12, or 24 rather than to bits 12, 24, or 0.

(4) The 36-bit main adder output is stored in a control register.

4.3.2. Double-Precision Fixed-Point Addition and Subtraction

The steps performed in the arithmetic section for the double-precision Add and Add Negative instructions are as follows:

(1) The 72-bit augend/minuend is transferred from two consecutive control registers to the 72-bit augend/minuend register of the arithmetic section.

(2) The full 72-bit addend/subtrahend is transferred from two main storage locations to the 72-bit addend/subtrahend register of the arithmetic section.

(3) The contents of the addend/subtrahend register are added to or subtracted from the contents of the augend/minuend register. The two phases of operation of the main adder proceed as explained in 4.2.

(4) The 72-bit main adder output is stored in two consecutive control registers.

4.3.3. Fixed-Point Overflow and Carry

In fixed-point arithmetic, the execution of certain instructions can result in an overflow or a carry condition. The overflow and carry conditions set bits D1 and D0 in the Processor State Register (PSR). These bits can be sensed by the action of certain other instructions.

Overflow designator D1 and carry designator D0 in the PSR are both affected by the execution of any of the instructions listed below. During the execution of these instructions, D1 and D0 are both set to zero. If an overflow condition occurs, D1 is set to 1; if a carry condition occurs, D0 is set to 1. Each of these designators remains set until the next time one of the instructions listed below is executed or until modified by execution of a Load Processor State Register instruction. The following instructions affect the carry and overflow designators:

- Add To A
- Add Negative To A
- Add Magnitude To A
- Add Negative Magnitude To A
- Add Upper
- Add Negative Upper
- Add To X
- Add Negative To X
- Double-Precision Fixed-Point Add
- Double-Precision Fixed-Point Add Negative

4.3.3.1. Overflow

An overflow condition occurs when the sum or difference produced by executing any of the instructions listed in 4.3.3 represents a number that is greater than the largest signed number that can be correctly represented in the given word length. This is of significance when the operands for an additive process are of the same sign or when the operands for a subtractive process have different signs. If overflow occurs in these cases, the sign of the result is unnatural. The condition of the overflow designator (D1 of PSR) can be tested by executing either the Jump Overflow or the Jump No Overflow instruction. Table 4–1 lists the sign bit combinations which would set D1 to a 1 bit indicating that an overflow has occurred.

OPERATION	INPUT OPERAND SIGN		RESULTANT SIGN
	AUGEND	ADDEND	
Addition	+	+	-
	-	-	+
	MINUEND	SUBTRAHEND	
Subtraction (Add Negative)	+	-	-
	-	+	+

Table 4–1. Sign Bit Combinations Which Set Overflow Designator

4.3.3.2. Carry

The carry designator (D0 of PSR) is set to 1 when an end-around carry occurs during the execution of an instruction listed in 4.3.3. The detection of a carry condition indicates that a carry was propagated out of the sign bit position and automatically added into the low-order bit position. The detection of the carry condition is significant when programming multiple-precision routines. In ones complement subtractive arithmetic, the carry condition can be equated to the no borrow condition, and the no carry condition to the borrow condition.

The condition of the carry designator can be tested by executing either the Jump Carry or Jump No Carry instructions. Table 4–2 lists the sign combinations for which the carry designator would be set to 1 indicating that a carry has occurred.

OPERATION	INPUT OPERAND SIGN		RESULTANT SIGN
	AUGEND	ADDEND	
Addition	+	-	+
	-	+	+
	-	-	+
	-	-	-
	MINUEND	SUBTRAHEND	
Subtraction (Add Negative)	+	+	+
	-	-	+
	-	+	+
	-	+	-

Table 4–2. Sign Bit Combinations Which Set Carry Designator

4.3.4. Fixed-Point Multiplication

The process of multiplying two single-precision fixed-point numbers consists of transferring the numbers to the arithmetic section, forming a 72-bit product by a series of main adder cycles and shifts, and transferring part or all of the product to either one or two control registers.

The various steps performed for the three fixed-point multiplication instructions are as follows:

(1) The two operands are transferred from storage to the arithmetic section:

- The multiplicand is transferred from a control register to a 36-bit multiplicand register.
- The multiplier is transferred from main storage, a control register, or the index subsection (j = 16_8 or 17_8) to a 36-bit multiplier register. If the multiplier is only a partial word, it is extended to a full 36-bit word as specified by the j field of the instruction.

(2) The leftmost bit of the multiplicand register and of the multiplier register are examined:

- If these bits are not identical, an indicator is set to indicate that the final product should be negative.
- If either operand is negative, it is complemented before the multiplication operation begins.

(3) A 72-bit accumulator register is cleared to +0.

(4) The multiplication algorithm compares two bits of the multiplier during each of the 18 cycles of the instruction. The appropriate multiple of the multiplicand is added to the upper half of the accumulator for each of the four possible combinations of multiplier bits (0, 1, 2, or 3). When the multiplier bits are 00 or 01, either zero or the multiplicand is added to the accumulator, and the result is right-shifted two bit positions. When the multiplier bits are 10, the partial product is right-shifted one bit position before the multiplicand is added in, and the resultant sum is then right-shifted one bit position. This process adds two times the multiplicand to the partial product. When the multiplier bits are 11, the multiplicand is subtracted from the partial product, and the resultant difference is right-shifted two bit positions. A one is then added to the next two multiplier bits. If this bit pair is 11, adding one changes it to 00, and subsequent bit pairs must be modified. This procedures adds 4 minus 1 to the partial product without requiring more than one full addition during each cycle. If the subtraction of the multiplicand from the partial product produces a negative result, the hardware automatically makes the necessary adjustments to shift bits from the negative upper half of the accumulator to its positive lower half.

(5) After the 18 cycles are completed, the final product is adjusted for correct sign.

(6) The final steps are dependent on which of the fixed-point multiply instructions is being performed:

- If the instruction is Multiply Integer, the contents of the accumulator are stored in two consecutive control registers.
- If the instruction is Multiply Single Integer, the 36 rightmost bits of the accumulator are stored in a control register. The 36 leftmost bits are lost.
- If the instruction is Multiply Fractional, the contents of the accumulator are shifted one bit position left circular and then stored in two consecutive control registers.

The one bit position left circular shift of the contents of the accumulator register performed just prior to storing the product produced for the Multiply Fractional instruction is the only difference between the multiply integer and multiply fractional processes.

For both multiply integer instructions: if each input operand has an implied binary point to the immediate right of its rightmost bit, then an implied binary point is located to the immediate right of the rightmost bit of the result.

For the Multiply Fractional instruction: if each input operand has an implied binary point between bits 35 and 34, then an implied binary point is located between bits 35 and 34 of the most significant word of the resultant product.

When it is desired to scale the operands as mixed numbers with the implied binary point located somewhere between or beyond the ends of the register, each number must be considered to be either a fraction or an integer multiplied by the appropriate power of two. The scaling associated with the resultant product can be determined from the algebraic laws of exponents, that is,

$$(x \cdot 2^n)\,(y \cdot 2^m) = (xy)2^{n+m}$$

For example, consider the following two 6-bit operands:

$3.5_{10} = 011.100$

$2.25_{10} = 010.010$

When multiplying these mixed numbers, they can be considered to be equivalent to the integers 011100 and 010010, where each integer has associated with it a factor of 2^{-3}, that is,

$011.100 = (011100)2^{-3}$

$010.010 = (010010)2^{-3}$

In integer arithmetic, the product of 011100 and 010010 would be 000111111000.

The scaled product of $(011100)2^{-3}$ and $(010010)2^{-3}$ is:

$(000111111000)2^{-6} = 000111.111000 = 7.875_{10}$

The two 6-bit operands could also be scaled as fractions:

$3.5_{10} = 011.100 = (0.11100)2^{2}$

$2.25_{10} = 010.010 = (0.10010)2^{2}$

In fractional arithmetic the product of 0.11100 and 0.10010 would be 0.01111110000.

The scaled product of $(0.11100)2^{2}$ and $(0.10010)2^{2}$ is:

$(0.01111110000)2^{4} = 00111.1110000 = 7.875_{10}$

4.3.5. Fixed-Point Division

The process of dividing one fixed-point number by another consists of loading the numbers into the arithmetic section, performing a series of trial subtractions to form a quotient and a remainder, transferring the properly signed quotient to a control register, and (for two of the three fixed-point divide instructions) transferring the properly signed remainder to the next higher addressed control register. The various steps performed in the arithmetic section for a fixed-point divide instruction are effectively as follows:

(1) The operands are transferred from storage to the arithmetic section:

- If the instruction is Divide Integer or Divide Fractional, the dividend is transferred from two consecutive control registers, and it fills the entire 72-bit dividend/remainder register.
- If the instruction is Divide Single Fractional, the dividend is transferred from a control register to the leftmost 36 bits of the 72-bit dividend/remainder register and the rightmost 36 bits of the dividend/remainder register are filled with bits identical to the sign bit of the dividend.
- The divisor is transferred from main storage, a control register, or the index subsection (j = 16_8 or 17_8) to a 36-bit divisor register. If the divisor is a partial word, it is extended to a full word as specified by the j field of the instruction.

(2) The leftmost bit of the 72-bit dividend/remainder register and that of the 36-bit divisor register are examined:

- If these bits are not identical, a negative quotient results.
- If the leftmost bit of the dividend/remainder register is a 1, a negative remainder results.
- When an operand is negative, the ones complement of the operand is placed in the register.

(3) The needed operating registers are prepared for subsequent operations and a trial subtraction and test are performed:

- A 36-bit quotient register is cleared to +0.
- If the instruction is Divide Integer, the 72-bit dividend/remainder register is left-shifted one bit position with the leftmost bit being discarded and a 0 bit inserted in the rightmost bit. No shifting occurs if the instruction is Divide Fractional or Divide Single Fractional.
- The contents of the divisor register are subtracted from the contents of the leftmost 36 bits of the dividend/remainder register. The difference is examined and then discarded.
  - If the leftmost bit of the difference is a 1 bit and the leftmost bit of the dividend/remainder register is a 0 bit, step 4 is initiated.
  - If the leftmost bit of the difference is a 0 bit, or if the leftmost bit of the dividend/remainder register is a 1 bit, a divide fault (see 4.3.6) is reported and the remaining steps are aborted. This occurs if the divisor register contains all 0 bits (the initial contents of the divisor register were +0 or -0) or if more than 36 bits are required to represent the numeric value of the quotient with its sign bit.

(4) A series of 35 shift operations, each followed by a trial subtraction, is used to form a positive quotient, as follows:

- For each shift operation, the contents of the 72-bit dividend/remainder register is left-shifted one bit position with the leftmost bit discarded and a 0 bit inserted in the rightmost bit position. The contents of the 36-bit quotient register are shifted in the same manner.
- Following each shift operation, a trial subtraction of the contents of the divisor register from the leftmost 36 bits of the 72-bit dividend/remainder register is performed.
- If the trial subtraction produces a negative result, the difference is discarded, and the contents of the quotient register remain unchanged.
- If the trial subtraction produces a positive result, the difference replaces the value in the leftmost 36 bits of the dividend/remainder register, and a 1 bit is stored in the rightmost bit of the quotient register.

(5) Following completion of the series of 35 shifts/trial subtractions, the quotient register contains the absolute value of the quotient, and bits 71 through 36 of the dividend/remainder register contain the absolute value of the remainder. The properly signed quotient and remainder are formed as follows:

- If the leftmost bits of the input operands are identical, the quotient register contains the properly signed positive quotient.
- If the leftmost bits of the input operands are not identical, the contents of the quotient register are complemented to form a negative quotient.
- If the leftmost bit of the input dividend was a 0 bit, bits 71 through 36 of the dividend/remainder register contain the properly signed positive remainder.
- If the leftmost bit of the input dividend was a 1 bit, the contents of the dividend/remainder register are complemented to form the properly signed negative remainder in bits 71 through 36.

(6) The final step in the arithmetic section depends on which fixed-point divide instruction is being performed:

- If the instruction is Divide Integer or Divide Fractional, the properly signed quotient and remainder are stored in two consecutive control registers.
- If the instruction is Divide Single Fractional, the properly signed quotient is stored in a control register. The remainder is not stored.

The one bit left shift of the dividend/remainder register performed for the Divide Integer instruction in step 3 is the only difference between the divide integer and divide fractional processes.

For the Divide Integer instruction, if each input operand has an implied binary point to the immediate right of its rightmost bit, then an implied binary point is located to the immediate right of the rightmost bit of the quotient and the remainder.

For both divide fractional instructions, if each input operand has an implied binary point to the immediate right of its rightmost bit, then an implied binary point is located between bits 35 and 34 of the quotient. For a Divide Fractional instruction, the implied binary point is located to the left of the 34^{th} bit of the 35-bit implied sign extension of the remainder.

Just as in multiplication, if the operands are to be scaled as mixed numbers, each number must be considered to be either a fraction or an integer multiplied by the appropriate power of two. The scaling associated with the resulting quotient can be determined from the algebraic laws of exponents, that is, $x \cdot 2^n / y \cdot 2^m = (x/y)2^{n-m}$.

4.3.6. Divide Fault

A divide fault condition is detected during the execution of a fixed-point divide instruction when an internal test determines that more than 36 bits are required to represent the properly signed quotient. A divide fault condition is also detected when any value is to be divided by +0 or -0 (fixed point) or by any floating-point number having a signed mantissa of +0 or -0. When a divide fault condition is detected, a divide fault interrupt occurs (see 8.3.2.5).

4.4. FLOATING-POINT ARITHMETIC

Floating-point arithmetic was designed to handle the scaling problems which arise in computations involving numbers which vary widely in range. In floating-point arithmetic, the numbers are represented in a special format so that the computer can automatically handle the scaling. The floating-point representation of a number consists of two parts. One part expresses the magnitude of the number; the other represents the size of the number within the range set off by the first part. This representation is very similar to scientific notation.

In scientific notation, the speed of light can be expressed as 1.863×10^5 miles per second. There are two parts to this expression: the power of ten which expresses the order of magnitude, and a number between one and ten which represents the size of the given quantity within the power of ten. The number of digits in this part of the expression indicates the accuracy of the measurement. The corresponding fixed-point representation for this number is 186300. It is also true that $1.863 \times 10^5 = 0.1863 \times 10^6 = 18.63 \times 10^4$; however, by convention, in scientific notation, the number is represented as a number between one and ten multiplied by the appropriate power of ten.

In floating-point notation, a number is represented as a binary fraction multiplied by the appropriate power of two. By convention, the binary point is generally placed so that the binary fraction represented is greater than or equal to 1/2 but less than 1.

4.4.1. Floating-Point Formats

Floating-point numbers in the UNIVAC 1108 system are represented in single-precision format as a 27-bit fractional quantity multiplied by the appropriate power of two, or in the double-precision format as a 60-bit fractional quantity multiplied by the appropriate power of two. The power of two is called the exponent. In machine representation, the exponents are biased to make them lie in the range of positive numbers or zero. These biased exponents are called characteristics. The fractional part is referred to as the mantissa.

Since the exponent base is always 2, it is not included in the actual machine representation of the floating-point number. The two parts represented are the biased exponent (characteristic) and the fractional part of the number (mantissa).

The single-precision format for a floating-point number is shown in Figure 4–1. The double-precision format for a floating-point number is shown in Figure 4–2.

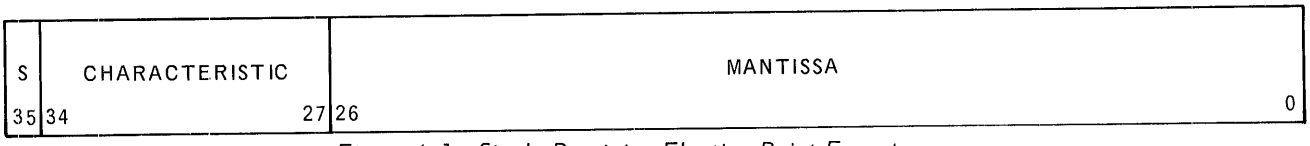


Figure 4–1. Single-Precision Floating-Point Format

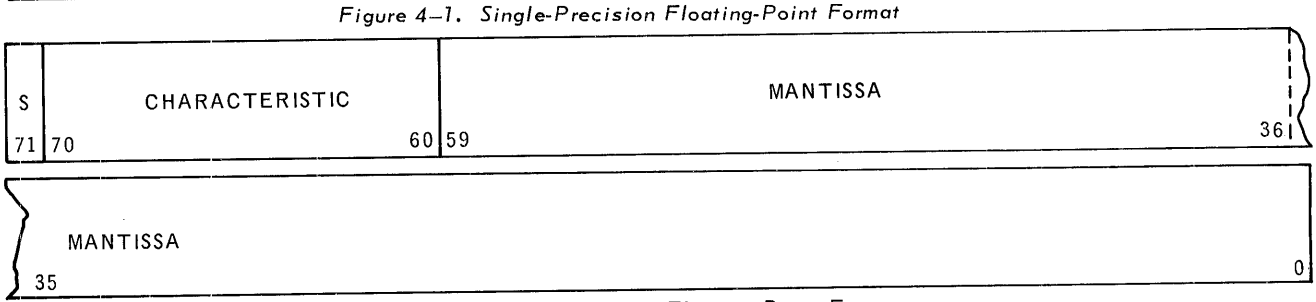


Figure 4–2. Double-Precision Floating-Point Format

- S = Sign

  The sign bit expresses the sign of the numerical quantity represented by the floating-point number.

  (1) If S = 0, the numerical quantity is positive (+).

  (2) If S = 1, the numerical quantity is negative (–).

- Characteristic

  The characteristic represents both the numerical value and the sign of the exponent.

  (1) Single-Precision Characteristic

  The 8-bit characteristic of a single-precision floating-point number represents an exponent value in the range +127 through –128. The characteristic is formed by adding a bias of +128 (200_8) to the exponent. Table 4–3 shows the range of characteristic values and corresponding exponent values.

DECIMAL VALUES		OCTAL VALUES	
CHARACTERISTIC	UNBIASED EXPONENT	CHARACTERISTIC	UNBIASED EXPONENT
255	+127	377	+177
128	000	200	000
000	–128	000	–200

Table 4–3. Single-Precision Floating-Point Characteristic Values vs. Exponent Values

  (2) Double-Precision Characteristic

  The 11-bit characteristic of a double-precision floating-point number represents an exponent value in the range +1023 through –1024. The characteristic is formed by adding a bias of +1024 (2000_8) to the exponent. Table 4–4 shows the range of characteristic values and the corresponding exponent values.

DECIMAL VALUES		OCTAL VALUES	
CHARACTERISTIC	UNBIASED EXPONENT	CHARACTERISTIC	UNBIASED EXPONENT
2047	+1023	3777	+1777
1024	0000	2000	0000
0000	–1024	0000	–2000

Table 4–4. Double-Precision Floating-Point Characteristic Values vs. Exponent Values

- Mantissa

The mantissa portion of a floating-point number represents the fractional part of the number. In the UNIVAC 1108, the fractional part is normalized so that the absolute values represented are greater than or equal to 1/2 but less than 1. Zero cannot be represented in this range and it is considered to be normalized as it stands. The binary point of a floating-point number is assumed to lie between the last bit of the characteristic and the first bit of the mantissa. The mantissa of a single-precision floating-point number contains 27 bits; for a double-precision floating-point number, the mantissa contains 60 bits. For any number that has more significant bits than can be contained in the mantissa, the excess bits are truncated.

4.4.1.1. Positive Single-Precision Floating-Point Numbers

A single-precision floating-point number can be derived from a positive decimal number as follows:

Example 1:

Given number = $+12_{10}$

$12_{10} = +1100_2 = .1100_2 \times 10_2^{+100_2}$

- Sign = + = 0
- Characteristic = exponent + bias

  $= 100_2 + 10\ 000\ 000_2$

  $= 10\ 000\ 100_2 = 204_8$

- Mantissa $= .1100_2$

  $= .110\ 000\ \ldots.\ 000_2 = .600\ 000\ 000_8$

The binary point to the left of the mantissa is not shown (it is implied) in the single-precision floating-point representation of the number $+12_{10}$, which is: $010\ 000\ 100\ 110\ 000\ \ldots.\ 000_2$ or 204600000000_8.

Example 2:

Given number = $+0.1875_{10}$

$0.1875_{10} = 0.0011_2 = .11 \times 10_2^{-10_2}$

- Sign = + = 0
- Characteristic = exponent + bias

  $= -10 + 10\ 000\ 000_2$

  $= 01\ 111\ 110_2 = 176_8$

- Mantissa $= .11_2$

  $= .110\ 000\ \ldots.\ 000_2 = .600\ 000\ 000_8$

The single-precision floating-point number for $+0.1875_{10}$ is:

$001\ 111\ 110\ 110\ 000\ \ldots.\ 000_2$, or

176600000000_8

4.4.1.2. Positive Double-Precision Floating-Point Numbers

A double-precision floating-point number can be derived from a given number as follows:

Example 1:

Given number = $+12_{10}$

$12_{10} = 1100_2 = .1100_2 \times 10_2{}^{+100_2}$

- Sign = + = 0
- Characteristic = unbiased exponent + bias

  $= 100_2 + 10\ 000\ 000\ 000_2$

  $= 10\ 000\ 000\ 100_2 = 2004_8$
- Mantissa = $.1100_2$

  $= .110\ 000\ \ldots\ 000_2 = .600\ \ldots\ 000_8$

The double-precision floating-point number for $+12_{10}$ is:

$010\ 000\ 000\ 100\ 110\ 000\ \ldots\ 000_2$, or $20046000\ldots000_8$

Example 2:

Given number = $+0.1875_{10}$

$0.1875 = 0.0011_2 = .11_2 \times 10_2{}^{-10_2}$

- Sign = + = 0
- Characteristic = unbiased exponent + bias

  $= 10_2 + 10\ 000\ 000\ 000_2$

  $= 01\ 111\ 111\ 110_2 = 1776_8$
- Mantissa = $.11_2$

  $= .110\ 000\ \ldots\ 000_2 = .600\ \ldots\ 000_8$

The double-precision floating-point number for $+0.1875_{10}$ is:

$001\ 111\ 111\ 110\ 110\ 000\ \ldots\ 000_2$, or $1776600\ \ldots\ 000_8$

4.4.1.3. Negative Floating-Point Numbers

A floating-point number can be derived to represent a given negative number as follows:

- Represent the given number as a positive floating-point number
- Form the ones complement of the entire positive floating-point number

Example 1:

Given number = -12_{10}

The single-precision floating-point number for $+12_{10}$ is $204\ 600\ 000\ 000_8$.

The single-precision floating-point number for -12_{10} is $573\ 177\ 777\ 777_8$.

Example 2:

Given number = -0.1875_{10}

The double-precision floating-point number for $+0.1875_{10}$ is $1776\ 600\ \ldots\ldots\ 0_8$.

The double-precision floating-point number for -0.1875_{10} is $6001\ 177\ldots\ldots 7_8$.

4.4.1.4. Residue

When a single-precision floating-point add or add negative operation is performed, the result consists of two single-precision floating-point numbers; one of the numbers represents the algebraic sum and the other number is the residue.

When the two 36-bit input operands for an Add or Add Negative instruction are transferred to the arithmetic section, their characteristics are examined, and the mantissa of the input operand with the smaller characteristic is right-shifted a number of bit positions equal to the difference between the characteristics; the bits shifted out of the 36-bit arithmetic register are saved in an auxiliary register. The portion of the mantissa saved in the auxiliary register is used to form the residue and it is not included in the algebraic addition. After completion of the addition and any shifting necessary to normalize the sum, the sum and the residue are packed into single-precision floating-point format and transferred to two consecutive control registers.

4.4.2. Normalized/Unnormalized Floating-Point Numbers

A floating-point number is normalized when the leftmost bit of the mantissa is not identical to the sign bit or when all bits of the mantissa are identical to the sign bit. A floating-point number is unnormalized when all bits of the mantissa are not sign bits and the leftmost bit of the mantissa is identical to the sign bit.

All floating-point operations produce a normalized result when the input operands are normalized.

The sums produced by Floating Add and Floating Add Negative instructions and the result produced by the Load And Convert To Floating instruction are always normalized regardless of whether or not the input operands are normalized. When either or both input operands are not normalized, the result obtained may be less accurate than if normalized input operands had been used.

Normalized input operands must be used for the Floating Multiply, Floating Divide, Compress And Load, and Expand And Load instructions. If normalized input operands are not used for these instructions, the results are undefined.

4.4.3. Floating-Point Characteristic Overflow/Underflow

Floating-point characteristic overflow/underflow occurs when the characteristic does not lie in the range representable in the number of bits allowed for the characteristic.

When any of the floating-point Add, Add Negative, Multiply, Divide, or Load And Convert instructions or the Compress And Load instruction are performed, overflow or underflow may occur.

4.4.3.1. Floating-Point Characteristic Overflow

Single-precision floating-point characteristic overflow occurs when the 8-bit characteristic of the resultant most significant single-precision floating-point word represents a number greater than 377_8 and the associated mantissa is not zero.

Double-precision floating-point characteristic overflow occurs when the 11-bit characteristic of the resultant double-precision floating-point number represents a number greater than 3777_8 and the associated mantissa is not zero.

When overflow is detected, an interrupt occurs which causes the CPU to execute the instruction in floating-point characteristic overflow fault interrupt location MSR+00246_8, rather than the next instruction in sequence. The initial contents of the A registers referenced by the instruction remain unchanged and the results of the floating-point operation are not stored.

4.4.3.2. Floating-Point Characteristic Underflow

Single-precision floating-point characteristic underflow occurs when the 8-bit characteristic of the resultant most significant single-precision floating-point word represents a negative number and the associated mantissa is not zero. This means that the exponent of the result is less than -200_8. If the characteristic of the residue (Floating Add, Floating Add Negative), remainder (Floating Divide), or the least significant single-precision word of the product (Floating Multiply) represents a negative number, this fact by itself does not result in underflow. Instead, the residue, remainder, or least significant word of the product is cleared to all 0 bits or set to all 1 bits (to reflect the appropriate sign).

When single-precision floating-point characteristic underflow occurs, an interrupt causes execution of the instruction in floating-point characteristic underflow fault interrupt location MSR+00245_8 rather than the next instruction in sequence. The initial contents of the A registers referenced by the instruction remain unchanged and the results of the floating-point operation are not stored.

Double-precision floating-point characteristic underflow occurs when the 11-bit characteristic of the result represents a negative number and the mantissa is not zero. This means that the exponent of the result is less than -2000_8.

When double-precision floating-point characteristic underflow is detected, the action performed by the CPU depends on the contents of the double-precision underflow designator (D5) of PSR.

- If D5 = 0 and underflow occurs, an interrupt causes execution of the instruction located in the floating-point characteristic underflow fault interrupt location MSR+00245$_8$. The initial contents of the A registers referenced by the instruction remain unchanged and the results of the floating-point operation are not stored.
- If D5 = 1 and underflow occurs, the interrupt does not occur. The result of the floating-point operation is stored as +0.

4.4.4. Mechanics of Floating-Point Arithmetic

The main adder is used in arithmetic processes involving the mantissas of floating-point numbers. For arithmetic processes involving the characteristics of floating-point numbers, a 9-bit characteristic adder is used for single-precision floating-point operations and a 12-bit characteristic adder for double-precision floating-point operations.

4.4.4.1. Floating-Point Addition

The process of adding two floating-point numbers consists of loading the numbers into the arithmetic section, using the characteristic adder to determine the difference between the characteristics of the two numbers, shifting the mantissa of the number having the smaller characteristic, adding the mantissas in the main adder, combining the results in floating-point format, and transferring the resulting floating-point numbers to control registers.

The input operands for floating-point addition need not be normalized numbers. For single-precision addition, the sum (most significant word produced) is always a normalized number. The residue word may or may not be a normalized number. For double-precision addition, the sum is always a normalized number.

4.4.4.1.1. Single-Precision Floating-Point Addition

The steps performed for single-precision floating-point addition are as follows:

(1) The operands (A) and (U) are transferred from storage to the arithmetic section.

(2) The operands are unpacked and the characteristic difference is determined. In the unpacking procedure, the 8-bit characteristics are separated from the mantissas and transferred to special characteristic registers. If an operand is negative, bits 34 through 27 are complemented during this transfer. One characteristic is subtracted from the other and the resultant difference is called the characteristic difference.

The sign of the characteristic difference indicates which of the two operands has the smaller characteristic, and its magnitude indicates how many bit positions the operand with the smaller characteristic should be shifted to the right in order to align it with the other operand. The larger characteristic is retained as the characteristic to be associated with the sum.

NOTE: The operands referred to in the discussion that follows are the unpacked operands in which sign bits replace the characteristic portion of the word.

(3) The operand having the smaller characteristic is transferred to the upper half of a 72-bit nonaddressable arithmetic accumulator and right-shifted to align it for addition.

- If the operand is positive, the lower half of the accumulator is cleared to +0; if the operand is negative, the lower half is set to -0.
- The accumulator is right-shifted, end off with sign fill on the left, a number of places equal to the magnitude of the characteristic difference. If this difference is greater than 63, the accumulator is shifted only 63 places.
- If the characteristic difference is zero, the two operands have equal characteristics. The operand in the A register is transferred to the upper half of the accumulator and no shifting takes place.

NOTE: In the discussions that follow, the bit positions of the nonaddressable 72-bit arithmetic accumulator are considered to be numbered as indicated in Figure 4–3.

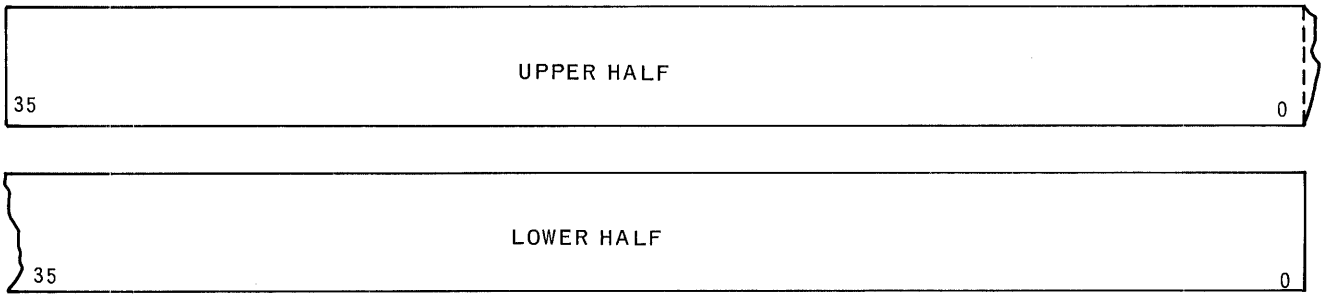


Figure 4–3. Nonaddressable 72-Bit Arithmetic Accumulator

(4) The operand having the larger characteristic is added to the upper half of the accumulator and the resultant sum replaces the previous contents of the upper half of the accumulator. Since the lower half of the accumulator is not involved in the addition, the sign of the upper half may, after the addition, differ from that associated with the lower half.

(5) The sum is normalized and both halves of the accumulator are packed.

- Whenever two normalized floating-point operands are added, the absolute value of the sum must fall between zero and two. Obviously, operands which can contain up to 27 significant bits could result in a sum containing 28 significant bits. During the unpacking operation, the characteristic portion was filled with sign bits and, after the addition, bits 35 through 28 of the sum are always sign bits. The first step of the normalization procedure is to check for a 28-bit sum. This is done by comparing bits 35 and 27 of the upper half of the accumulator.

- Case I. If bit 35 = bit 28 the sum contains 27 significant bits or less. The residue from the alignment (step 3), which is stored in the lower half of the accumulator, is packed first. This consists of right-shifting the residue nine bit positions to make room for the 8-bit characteristic and the mantissa sign bit. The residue characteristic is obtained by subtracting 27 from the characteristic associated with the sum. A special case arises if this difference is a negative value (see 4.4.4.1.2). The sign of the input operand which was originally transferred to the upper half of the accumulator (before the addition) is retained as the sign to be associated with the residue word. If this sign bit is a 1 bit, the characteristic is complemented when it is transferred into the characteristic portion of the word. When the sign bit is a 0 bit, complementation is omitted and the characteristic is packed directly into the characteristic portion of the word. After the residue word is packed, the sum stored in the upper half of the accumulator is normalized. The normalization is accomplished by comparing bits 27 and 26. If they are not equal, the sum contains 27 bits and it is already normalized. If bits 27 and 26 are equal, the upper half of the accumulator is left circular shifted (applies only to upper half) until bits 27 and 26 are unequal. Sign bits are brought into the right end during the shifting and this shifting terminates after a maximum of 27 shifts. The characteristic of the sum is decreased by a number equal to the number of places the sum was shifted to achieve normalization and then a check is made for floating-point characteristic underflow. (See special cases, 4.4.4.1.2.) If there is no underflow, the characteristic is packed with the sum. If the sum is negative, the characteristic is complemented as it is transferred to the word. If the sum is positive, the complementation is omitted.

- Case II. If bit 35 is not equal to bit 27 the addition produced a 28-bit sum and the entire 72-bit accumulator is right-shifted one bit position to achieve normalization. The characteristic of the sum is increased by one and checked for floating-point characteristic overflow. (See special cases, 4.4.4.1.2.) This step occurs before either the upper or lower half of the accumulator is packed. If there is no overflow, the residue word in the lower half of the accumulator is right-shifted nine bit positions. The residue characteristic is obtained by subtracting 27 from the characteristic of the sum. The resultant characteristic is adjusted for sign (complemented if the sum is negative) and packed into the characteristic portion of the residue word. (See special cases, 4.4.4.1.2.) The residue in the lower half of the accumulator can be considered to be an extension of the sum formed in the upper half because a 28-bit sum can only occur when operands of the same sign are added. After packing the residue word, the characteristic of the sum is adjusted for sign (complemented if sum is negative) and transferred to the characteristic portion of the sum in the upper half of the accumulator.

(6) Finally, the packed floating-point sum in the upper half of the accumulator is transferred to the A register specified by the instruction word. The packed residue in the lower half of the accumulator is transferred to the next consecutive A register, A+1.

4.4.4.1.2. Single-Precision Floating-Point Addition – Special Cases

The following paragraphs describe the manner in which floating-point characteristic overflow/underflow and zero-resultant mantissas are handled by the CPU.

(1) Floating-Point Characteristic Overflow/Underflow

If floating-point characteristic overflow is detected during the normalization process (Case II, 4.4.4.1.1), the normal procedure is aborted and an interrupt occurs as indicated in 4.4.3.1. If underflow is detected (Case I, 4.4.4.1.1), the normal procedure is aborted and an interrupt occurs as indicated in 4.4.3.2. Overflow or underflow is never reported if the mantissa is zero; see (2) below.

When overflow or underflow is detected and an interrupt occurs, the contents of the 72-bit arithmetic accumulator are lost and the input operands remain in the specified A and U.

If underflow occurs on the residue characteristic, floating-point underflow is not reported. Depending upon the sign of the input operand, either +0 or -0 is transferred to register A+1, where A denotes the A register specified by the Floating Add or Floating Add Negative instruction.

(2) Zero Mantissa

Bit D8 of the PSR specifies whether or not a characteristic is to be attached to the result when a zero mantissa is generated.

If D8 = 0, the characteristic is cleared to 0 bits whenever a zero mantissa is produced. If the mantissa is -0, the entire word is cleared to +0.

If D8 = 1, the characteristic is formed in the normal manner and packed into the characteristic portion of the word. The procedure followed is that described for Case I in 4.4.4.1.1. When the mantissa is zero, the normalization step requires 27 shifts before it terminates, and the characteristic attached to the sum will be 27 less than the characteristic of the larger input operand. If the mantissa is -0, the characteristic is complemented.

Whenever there is floating-point characteristic overflow or underflow and the mantissa is zero, no overflow or underflow interrupt is generated, and the entire word is cleared to +0 (regardless of the value of D8).

4.4.4.1.3. Double-Precision Floating-Point Addition

The steps performed for double-precision floating-point addition are as follows:

(1) The two operands are transferred from storage to the arithmetic section:

- The first operand is transferred from two consecutive control registers to a 72-bit register.
- The second operand is transferred from two main storage locations to a different 72-bit register.

(2) The operands are unpacked and the characteristic difference is determined:

- The characteristics of the two operands are transferred to two 11-bit characteristic registers. Bits 71 through 60 are complemented during the transfer if the corresponding mantissa is negative.
- The characteristic portion of each operand is filled with sign bits.
- The characteristic difference is formed by subtracting the characteristic of the first operand from the characteristic of the second operand.
- If the difference is positive or zero, the first operand is transferred to the 72-bit augend register, and the second operand is transferred to the 72-bit addend register.
- If the difference is negative, the first operand is transferred to the addend register, and the second operand is transferred to the augend register. The transferred operands are the unpacked operands in which sign bits replace the characteristic portion of the double-precision floating-point word.
- The absolute value of the characteristic difference is retained for the addition alignment shift count.
- The larger characteristic is retained as the characteristic to be associated with the sum.

(3) The contents of the augend register are right-shifted end off with sign fill on the left. The shift count is the absolute value of the characteristic difference.

- If the characteristic difference is zero, no shifting occurs.
- If the characteristic difference is greater than 60, the contents of the augend register are shifted only 60 bit positions.

(4) The contents of the addend register are added to the contents of the augend register and the resultant sum replaces the previous contents of the augend register. The leftmost 11 bits of the sum are always sign bits.

(5) The sum is normalized and packed.

- Since the input mantissas can contain no more than 60 significant bits, the sum can have no more than 61 significant bits. The first step of the normalization procedure is to check for a 61-bit sum. This is done by comparing bits 71 and 60. If they are equal, the sum contains 60 or less significant bits. If they are unequal, the sum contains 61 significant bits.
- When the sum contains 61 significant bits, the augend register is right-shifted one bit position and the characteristic associated with the sum (last item in step 2) is incremented by one. This characteristic is examined for floating-point characteristic overflow. Overflow can occur only if the characteristic was 3777_8 before the incrementation. If overflow occurs, the normal procedure is aborted and an interrupt occurs as indicated in 4.4.3.1. If no overflow occurs, the characteristic is transferred to the characteristic portion of the augend register. If the sum is negative, it is complemented during the transfer.

- When the sum is not zero and contains 60 significant bits or less, bits 71 and 59 are examined. If bit 71 does not equal bit 59, the sum is already normalized. The characteristic is adjusted for sign and transferred to the characteristic position of the augend register. If bits 71 and 59 are equal, the augend register is shifted left circularly until bit 71 does not equal bit 59. The characteristic is decremented by a number equal to the number of left shifts and checked for floating-point characteristic underflow. If there is no underflow, the characteristic is transferred to the characteristic portion of the augend register. It is complemented during the transfer if the sum is negative. In double-precision operations, bit D5 of the PSR specifies whether or not the floating-point underflow interrupt will occur in the event of underflow. The special procedures which now take place are discussed in 4.4.3.2.

- If the sum is zero, the augend register is cleared to +0 and stored in the specified control registers.

(6) The contents of the 72-bit augend register are stored in the two consecutive control registers, specified in the Double-Precision Add instruction.

4.4.4.2. Floating-Point Add Negative (Subtraction)

Floating-point subtraction (both single precision and double precision) is performed following the same procedure as is followed for floating-point addition except that the ones complement of the subtrahend specified by the instruction word is used as the second input operand.

The input operands for floating-point subtraction need not be normalized numbers. For single-precision subtraction, the difference (most significant result word) is always a normalized number. The residue word may or may not be a normalized number. For double-precision subtraction, the difference is always a normalized number.

4.4.4.3. Floating-Point Multiplication

The process of multiplying two floating-point numbers consists of sending the operands to the arithmetic section, unpacking, multiplying the mantissas, adding the characteristics, packing the results into floating-point format, and transferring the resultant floating-point words to control registers. The sign of the product is determined by the algebraic rules for signs except as specified in 4.6. The following explanations apply specifically to all cases in which both input operands are normalized numbers. The explanations apply in general to those cases in which either or both operands are not normalized numbers. The results obtained for all cases in which either or both operands are not normalized numbers are undefined.

4.4.4.3.1. Single-Precision Floating-Point Multiplication

The steps performed for single-precision floating-point multiplication are as follows:

(1) The two operands are transferred from storage to the arithmetic section:

- The multiplicand is transferred from a control register to the 36-bit multiplicand register.
- The multiplier is transferred from main storage or a control register to the 36-bit multiplier register.

(2) The sign bit of each operand is examined to determine the sign of the product to be produced and then positive operands are provided.

- If the operands have identical signs, the sign of each word of the product will be positive.
- If the operands do not have identical signs, the sign of each word of the product will be negative, except that a zero product is always represented as +0.
- If an input operand is negative, it is complemented to provide positive operands.

(3) The two positive operands are unpacked to separate the characteristics from the mantissas, the characteristics are saved for use in step 6, bits 34 through 27 of both the multiplicand and the multiplier registers are cleared to 0 bits, and a 72-bit accumulator register is cleared to +0.

(4) The product of the mantissas of the two positive operands is formed as follows:

- The contents of the multiplicand register are multiplied by the contents of the rightmost 28 bits of the multiplier register using only 14 repetitions of step 4 of 4.3.4. The product is accumulated in the 72-bit accumulator register.
- The contents of the accumulator are left-shifted one bit position (end off, with 0-bit fill on the right). The 54-bit product of the 27-bit mantissas of the two positive operands is now in bits 62 through 9 of the accumulator. Bits 71 through 63 and 8 through 0 of the accumulator contain 0 bits.

(5) Bit 62 of the accumulator is used to control initiation of a normalizing shift which ensures that the most significant word of the product will be normalized if normalized operands were used.

- If bit 62 contains a 1 bit, no normalizing shift is performed.
- If bit 62 contains a 0 bit, the entire 72-bit accumulator is shifted one bit position left circular and this step is repeated until bit 62 contains a 1 bit.

(6) Characteristic arithmetic is performed using the 9-bit characteristic adder. Tests for floating-point characteristic overflow and underflow are performed.

- The characteristics of the two positive operands saved from step 3 are added to produce the characteristic sum, which is biased by 400_8.
- If a normalizing shift was not performed in step 5, 200_8 is subtracted from the characteristic sum to produce the characteristic for the most significant product word and 233_8 is subtracted from the characteristic sum to produce the characteristic for the least significant product word.
- If a normalizing shift was performed in step 5, 201_8 is subtracted from the characteristic sum to produce the characteristic for the most significant product word and 234_8 is subtracted from the characteristic sum to produce the characteristic for the least significant product word.
- If the mantissa of the most significant product word is not zero and the characteristic for the most significant product word represents a number greater than 377_8, overflow (see 4.4.3.1) is reported, and all following steps are aborted.
- If the mantissa of the most significant product word is not zero and the characteristic for the most significant product word represents a negative number, underflow (see 4.4.3.2) is reported, and all following steps are aborted.

(7) A single-precision floating-point word representing the most significant word of the product is packed in the upper half of the accumulator:

- The characteristic from step 6 is stored in bits 70 through 63 of the accumulator.
- If the input operands were oppositely signed, the upper half of the accumulator is complemented (product developed by multiply algorithm is always positive).
- If the mantissa is zero, the resultant word is controlled by D8 of the PSR. If D8 = 0, the upper half of the accumulator is cleared to +0. If D8 = 1, the word remains packed.
- If there is overflow or underflow for the most significant product word and the corresponding mantissa is zero, the floating-point characteristic underflow or overflow interrupt does not occur. If D8 = 0, both the characteristic and mantissa are set to +0. If D8 = 1, the characteristic is set to either +0 or -0, whichever agrees with the sign of the mantissa.

(8) A single-precision floating-point word representing the least significant product word is formed in bits 35 through 0 (lower half) of the accumulator.

- The lower half of the accumulator is right-shifted nine bit positions (end off with 0-bit fill in bits 35 through 27) so as to place the 27 low-order bits of the normalized product (from steps 4 and 5) in bits 27 through 0 of the accumulator.
- The 8-bit characteristic of the least significant product word is stored in bits 34 through 27 of the accumulator. If there is underflow, the lower half of the accumulator is cleared to +0. (See 6.5.5 for the special case of overflow on the least significant product word.)

- If the signs of the original operands were not identical, the lower half of the accumulator is complemented.
- The value of D8 has no effect on the least significant product word.

(9) The most significant product word (developed in step 7) and the least significant product word (developed in step 8) are stored in two consecutive control registers.

4.4.4.3.2. Double-Precision Floating-Point Multiplication

The steps performed for double-precision floating-point multiplication are as follows:

(1) The two operands are transferred from storage to the arithmetic section.

- The multiplicand is transferred from two consecutive control registers to the 72-bit multiplicand register.
- The multiplier is transferred from two main storage locations to the 72-bit multiplier register.

(2) The sign bit of each operand is examined to determine the sign of the product to be produced, and positive operands are then provided.

- If the operands have identical signs, the sign of the product will be positive.
- If the operands do not have identical signs, the sign of the product will be negative except that a zero product is always represented as +0.
- If an operand is negative, it is complemented to provide a positive operand.

(3) The two positive operands are unpacked to separate the characteristics from the mantissas, the characteristics are saved for use in subsequent characteristic arithmetic, bits 70 through 60 of both the multiplicand register and the multiplier register are cleared to 0 bits, and a 72-bit accumulator register is cleared to +0.

(4) The product of the mantissas of the two positive operands is formed. The contents of the multiplicand register are multiplied by the contents of bits 59 through 0 of the multiplier register. A series of 30 major cycles is required. Each major cycle includes a 72-bit main adder cycle and a shift of the contents of the accumulator and multiplier registers.

- The inputs for each main adder cycle are the contents of the accumulator, and either the contents of the 72-bit multiplicand register or a double-precision word of +0. The main adder output replaces the contents of the accumulator.
- The procedure for each main adder cycle and the associated shifting of the accumulator is dependent on the contents of the two rightmost bits of the multiplier register as explained in step 4 of 4.3.4 on fixed-point multiplication. When the multiplier bits which control the 30^{th} cycle are 11, the iteration must be carried one step further. This final step consists of adding the contents of the multiplicand register to the contents of the accumulator with no subsequent shifting of the accumulator.

- The final contents of the accumulator are 0 bits in bits 71 through 60; the 60 most significant bits of the product are located in bits 59 through 0. The 60 least significant bits of the 120-bit product are discarded.

(5) Bit 59 of the accumulator is used to control initiation of a normalizing shift to ensure that the result of the multiplication will be a normalized double-precision floating-point number if normalized input operands were used. If bit 59 is a 0 bit, the accumulator is shifted one bit position left circularly.

(6) Characteristic arithmetic is performed using the 12-bit characteristic adder. Tests for floating-point characteristic overflow and underflow and other special cases are performed.

- The characteristics of the two positive operands saved from step 3 are added to produce the characteristic sum which is biased by 4000_8.
- If a normalizing shift was not performed in step 5, 2000_8 is subtracted from the characteristic sum to produce the characteristic for the product.
- If a normalizing shift was performed in step 5, 2001_8 is subtracted from the characteristic sum to produce the characteristic for the product.
- If the mantissa is not zero and the characteristic for the product represents a number greater than 3777_8, overflow (see 4.4.3.1) is reported, and all following steps are aborted.
- If the mantissa is not zero and there is underflow, D5 of the PSR determines which action is performed next. If D5 = 0, the floating-point characteristic underflow interrupt occurs and all following steps are aborted. If D5 = 1, the 72-bit accumulator is cleared to +0 and step 8 is initiated to store the product.
- If the accumulator contains all 0 bits, the characteristic of the product is discarded, and step 8 is initiated to store the product as +0 regardless of the signs of the input operands.

(7) If none of the special cases discussed in step 6 are detected, a double-precision floating-point number is developed in the accumulator which reflects the signs of the input operands.

- The 11-bit characteristic for the product developed in step 6 is stored in bits 71 through 60 of the accumulator to form a positive floating-point number.
- If the signs of the input operands differ, the accumulator contents are complemented.

(8) The contents of the accumulator are stored in two consecutive control registers.

4.4.4.4. Floating-Point Division

The process of dividing one floating-point number by another consists of loading the numbers into the arithmetic section, subtracting characteristics, dividing one mantissa by the other, combining the results into floating-point format, and transferring the result to the specified control registers. The sign of the quotient is determined by the algebraic rules for signs except as specified in 4.6. The following explanations apply specifically to all cases in which both input operands are normalized numbers. The explanations apply in general to those cases in which either or both operands are not normalized numbers. For all cases in which either or both operands are not normalized numbers, the results are undefined.

4.4.4.4.1. Single-Precision Floating-Point Division

The steps performed for single-precision floating-point division are as follows:

(1) The two operands are transferred from storage to the arithmetic section.

- The dividend is transferred from a control register to a 36-bit dividend/remainder register.
- The divisor is transferred from main storage or a control register to a 36-bit divisor register.

(2) The sign bit of each operand is examined to determine the sign of the quotient and the sign of the remainder. Positive operands are provided.

- If the sign bits are identical, the sign of the quotient will be positive.
- If the sign bits are not identical, the sign of the quotient will be negative. Zero results are the only exception (represented as +0).
- If the leftmost bit in the dividend register is a 0 bit, the sign of the remainder will be positive; if it is a 1 bit, the sign of the remainder will be negative.
- If an input operand is negative, it is complemented.

(3) The two positive operands are unpacked to separate the characteristics from the mantissas, the characteristics are saved for use in subsequent characteristic arithmetic, bits 34 through 27 of the dividend/remainder and the divisor registers are cleared to 0 bits, and a 36-bit quotient register is cleared to all +0. If the divisor register now contains all 0 bits, a divide fault (see 4.3.6) is reported, and all remaining steps are aborted.

(4) The division process is performed by a series of 27 or 28 main adder cycles. In each main adder cycle, the contents of the 36-bit divisor register are subtracted from the contents of the 36-bit dividend/remainder register.

- If an end-around borrow is not generated on the first main adder cycle, 26 more cycles are required.
- If an end-around borrow is generated on the first main adder cycle, 27 more cycles are required.

- For each main adder cycle, absence of an end-around borrow signifies that the number in the dividend/remainder register is greater than or equal to the number in the divisor register. In this instance, the main adder output replaces the contents of the dividend/remainder register and a 1 bit is stored in the rightmost bit of the quotient register.
- For each main adder cycle, occurrence of an end-around borrow signifies that the number in the dividend/remainder register is less than the number in the divisor register. In this case, the main adder output is discarded and no change is made to the rightmost bit position of the quotient register.
- Following each main adder cycle except the last, the contents of the dividend/remainder register and the contents of the quotient register are shifted one bit position left circularly to prepare for the next main adder cycle.
- Following the completion of the last main adder cycle and the storing of any results, the rightmost 27 bits of the quotient register contain the mantissa for the quotient and the rightmost 27 bits of the dividend/remainder register contains the mantissa for the remainder.

(5) Characteristic arithmetic is performed using the 9-bit characteristic adder. Tests for floating-point characteristic overflow and underflow are performed.

- The characteristic associated with the divisor is subtracted from the characteristic associated with the dividend. This difference does not include a bias.
- If an end-around borrow occurred in the first main adder cycle of step 4, 200_8 is added to the characteristic difference to produce the characteristic for the quotient and 33_8 is subtracted from the characteristic of the dividend to produce the characteristic for the remainder.
- If an end-around borrow did not occur in the first main adder cycle of step 4, 201_8 is added to the characteristic difference to produce the characteristic for the quotient and 32_8 is subtracted from the characteristic of the dividend to produce the characteristic for the remainder.
- If the quotient register does not contain all 0 bits and the characteristic for the quotient represents a number greater than 377_8, overflow (see 4.4.3.1) is reported, and all following steps are aborted.
- If the quotient register does not contain all 0 bits and the characteristics for the quotient represents a negative number, underflow (4.4.3.2) is reported, and all following steps are aborted.

(6) The quotient is developed in single-precision floating-point format in the quotient register as follows:

- The quotient mantissa occupies bits 26 through 0 of the quotient register.
- The 8-bit quotient characteristic developed in step 5 is stored in bits 35 through 27 of the quotient register.

- If the signs of the input operands are not identical, the contents of the quotient register are complemented.
- If the mantissa of the quotient is zero, the next operation is controlled by D8 of the PSR. If D8 = 0, the quotient register is cleared to +0 and step 7 is initiated. If D8 = 1, step 6 is performed without any consideration of the mantissa. If there is overflow or underflow when D8 = 1 and the mantissa is zero, the characteristic is cleared to +0 or -0, whichever agrees with the sign of the mantissa.

(7) The remainder is developed in single-precision floating-point format in the 36-bit dividend/remainder register as follows:

- The remainder mantissa occupies bits 26 through 0 of the dividend/remainder register.
- If the 8-bit remainder characteristic represents a negative number, the dividend/remainder register is cleared to +0; if it does not represent a negative number, it is stored in bits 35 through 27 of the dividend/remainder register.
- If the input dividend operand was negative, the contents of the dividend/remainder register are complemented.

(8) The contents of the 36-bit quotient register and the contents of the 36-bit dividend/remainder register are stored in two consecutive control registers.

4.4.4.4.2. Double-Precision Floating-Point Division

The steps performed for double-precision floating-point division are as follows:

(1) The two operands are transferred from storage to the arithmetic section.

- The dividend is transferred from two consecutive control registers to a 72-bit dividend register.
- The divisor is transferred from two main storage locations to a 72-bit divisor register.

(2) The sign bit of each operand is examined to determine the sign of the quotient. Positive operands are provided.

- If the sign bits are identical, the sign of the quotient will be positive.
- If the sign bits are different, the sign of the quotient will normally be negative, except that zero quotients are always represented as +0.
- If an operand is negative, the operand is complemented.

(3) The two positive operands are unpacked to separate the characteristics from the mantissas, the characteristics are saved for use in subsequent characteristic arithmetic, bits 70 through 60 of the dividend and the divisor registers are cleared to 0 bits, and a 72-bit quotient register is cleared to +0. If the divisor register now contains all 0 bits, a divide fault (see 4.3.6) is reported, and all remaining steps are aborted.

(4) The division process is performed by a series of 60 or 61 main adder cycles. In each cycle, the contents of the 72-bit divisor register are subtracted from the contents of the 72-bit dividend register.

- If an end-around borrow is generated on the first cycle, 60 additional cycles are required.
- If an end-around borrow is not generated on the first cycle, 59 additional cycles are required.
- For each main adder cycle, the absence of an end-around borrow signifies that the number in the dividend register is greater than or equal to the number in the divisor register. In this instance, the main adder output replaces the contents of the dividend register, and a 1 bit is stored in the rightmost bit position of the quotient register.
- For each main adder cycle, the occurrence of an end-around borrow signifies that the number in the dividend register is less than the number in the divisor register. In this case, the main adder output is discarded and no change is made to the rightmost bit position of the quotient register.
- Following each main adder cycle, except the last, the contents of the dividend register and the contents of the quotient register are shifted one bit position left circularly to prepare for the next main adder cycle.
- At the completion of the last main adder cycle (and the storing of results if appropriate), the rightmost 60 bits of the quotient register contain the mantissa for the absolute value of the quotient.

(5) Characteristic arithmetic is performed using the 12-bit characteristic adder. Tests for floating-point characteristic overflow and underflow, and other special cases are performed.

- The difference between the characteristics of the two positive operands saved from step 3 is found by subtracting the characteristic of the divisor from the characteristic of the dividend. This difference does not include the bias.
- If the quotient mantissa is zero, the characteristic of the quotient is ignored and step 7 is initiated to store all 0 bits regardless of the signs of the input operands.
- If an end-around borrow was not generated in the first main adder cycle of step 4, 2001_8 is added to the characteristic difference to produce the characteristic for the quotient.
- If an end-around borrow was generated in the first main adder cycle of step 4, 2000_8 is added to the characteristic difference to produce the characteristic for the quotient.
- If the quotient mantissa is not zero and the characteristic for the quotient represents a number greater than 3777_8, overflow (see 4.4.3.1) is reported, and all following steps are aborted.

- If the quotient mantissa is not zero and the characteristic of the quotient represents a negative number, the next operation is controlled by D5 of the PSR. If D5 = 0, underflow (see 4.4.3.2) is reported, and all following steps are aborted. If D5 = 1, the quotient register is cleared to +0 and step 7 is initiated to store all 0 bits regardless of the signs of the input operands.

(6) If none of the special cases discussed in step 5 were detected, a double-precision floating-point number is developed in the quotient register, which reflects the signs of the input operands.

- The 11-bit characteristic for the quotient developed in step 5 is stored in bits 70 through 60 of the quotient register to form a positive floating-point number.
- If the signs of the input operands differ, the contents of the quotient register are complemented.

(7) The contents of the quotient register are stored in two consecutive control registers.

4.5. CONVERTING A FIXED-POINT NUMBER TO A FLOATING-POINT NUMBER

Conversion of a fixed-point number to floating-point number is performed in the arithmetic section. The first input operand contains a characteristic (biased exponent) which defines the location of the binary point for the fixed-point number with respect to the standard position of the binary point for a floating-point number. The second input operand is the signed fixed-point number to be converted.

The conversion process consists of transferring the two operands to the arithmetic section, shifting the fixed-point number, if necessary, to position its bits as the mantissa for a normalized floating-point number, modifying the characteristic to reflect the magnitude and direction of the normalizing shift, packing the shifted fixed-point number (the mantissa) and the modified characteristic in floating-point format, and loading the packed results in a control register (conversion to single-precision floating-point format) or into two consecutive control registers (conversion to double-precision floating-point format).

4.5.1. Conversion To Single-Precision Floating-Point Format

The Load And Convert To Floating instruction is used to convert a single-precision fixed-point number to single-precision floating-point format. The steps performed for this instruction are as follows:

(1) The two operands are transferred from storage to the arithmetic section.

- The first operand is transferred from a control register to the arithmetic section. The rightmost eight bits of the operand are stored in a 9-bit characteristic register which contains a 0 bit in its leftmost bit position. The leftmost 28 bits of the operand are ignored.
- The second operand (signed fixed-point number) is transferred from main storage or a control register to a 36-bit operating register.

(2) The sign of the fixed-point number is examined to determine the expected sign of the result. If the fixed-point number is negative, the contents of the operating register are complemented.

(3) The positive fixed-point number in the operating register is then examined. The contents of the operating register are shifted, if necessary, to normalize the mantissa, and a shift count is determined.

- If the operating register contains more than 27 significant bits, it is right-shifted (end off with zerofill on the left) to move the most significant bit to bit 26. The number of bit positions shifted is the shift count. This number varies from one to eight.
- If the operating register contains exactly 27 significant bits, no normalizing shift is needed, and the shift count is zero.
- If the operating register contains less than 27 significant bits, it is left-shifted (end off with zerofill on the right) to move the most significant bit to bit 26. The shift count can vary from one to 26.
- If the operating register contains +0, the shift count is 27.

(4) The shift count developed in step 3 is either added to or subtracted from the contents of the 9-bit characteristic register using the 9-bit characteristic adder. Addition is performed if a right shift was required in step 3; otherwise, subtraction is performed. The characteristic adder output (result characteristic) is stored in bits 34 through 27 of the operating register. The special cases are handled as follows:

- If the result characteristic represents a number greater than 377_8, overflow (see 4.4.3.1) is reported, and all following steps are aborted.
- If the result characteristic represents a negative number and the mantissa is not all 0 bits, underflow (see 4.4.3.2) is reported, and all following steps are aborted.
- If bits 26 through 0 of the operating register are all 0 bits and D8 of the PSR equals 0, bits 35 through 27 of the register are cleared to all 0 bits, and step 6 is initiated to store a result consisting of all 0 bits.
- If bits 26 through 0 of the operating register are all 0 bits, D8 = 1, and the result characteristic represents a negative number; bits 35 through 27 of the operating register are cleared to all 0 bits and step 5 is initiated to consider the sign of the result. There is no special treatment of the result if bits 26 through 0 of the operating register are all 0 bits, D8 = 1, and the result characteristic represents a positive number.

(5) If the input fixed-point number was negative, the contents of the operating register are complemented to obtain the properly signed result.

(6) The contents of the operating register are stored in a control register.

4.5.2. Conversion To Double-Precision Floating-Point Format

The Double Load And Convert To Floating instruction is used to convert a double-precision fixed-point number to double-precision floating-point format. The steps are as follows:

(1) The two operands are transferred from storage to the arithmetic section.

- The first operand is transferred from a control register to the arithmetic section. The rightmost 11 bits of the operand are stored in a 12-bit characteristic register which contains a 0 bit in its leftmost bit position. The leftmost 25 bits of the operand are ignored.
- The second operand, a signed double-precision fixed-point number, is transferred from two main storage locations to a 72-bit operating register.

(2) The sign of the fixed-point number is examined to determine the expected sign of the result. If the fixed-point number is negative, the contents of the operating register are complemented.

(3) The positive fixed-point number in the operating register is then examined. The contents of the operating register are shifted, if necessary, to normalize the mantissa, and a shift count is determined.

- If the operating register contains more than 60 significant bits, it is right-shifted (end off with zero fill on the left) to move the most significant bit to bit 59. The number of bit positions shifted is the shift count. This number can vary from one to 11.
- If the operating register contains 60 significant bits, no normalizing shift is needed and the shift count is zero.
- If the operating register contains less than 60 significant bits, it is left-shifted (end off with zero fill on the right) to move the most significant bit to bit 59. The shift count can vary from one to 59.
- If the operating register contains +0, step 6 is initiated to store a result consisting of all 0 bits.

(4) The shift count developed in step 3 is either added to or subtracted from the contents of the 12-bit characteristic register using the 12-bit characteristic adder. Addition is performed if a right shift was required in step 3; otherwise, subtraction is performed. The characteristic adder output (result characteristic) is stored in bits 70 through 60 of the operating register. The special cases are handled as follows:

- If the result characteristic represents a number greater than 3777_8, overflow (4.4.3.1) is reported, and all following steps are aborted.
- If the result characteristic represents a negative number and D5 of the PSR equals zero, underflow is reported (4.4.3.2), and all following steps are aborted.
- If the result characteristic represents a negative number and D5 = 1, underflow is not reported. Instead, the operating register is cleared to +0 and step 6 is initiated to store a result consisting of all 0 bits.

(5) If the input fixed-point number was negative, the contents of the operating register are complemented to form the properly signed result.

(6) The contents of the 72-bit operating register are stored in two consecutive control registers.

4.6. FLOATING-POINT ZERO – SUMMARY

A single-precision floating- point word is *floating-point zero* if bits 35 and 26 through 0 of the word contain all 0 bits (+0) or all 1 bits (-0).

A double-precision floating-point number is floating- point zero if bits 71 and 59 through 0 (bits 35 and 23 through 0 of the high order word and all 36 bits of low order word) contain all 0 bits or all 1 bits.

Thus, floating- point zero can be defined as a floating-point number having all mantissa bits identical to the sign bit.

4.6.1. Single-Precision Floating-Point Zero

Six of the floating-point instructions produce results in single- precision floating- point format. These instructions and the names used to identify the results for each instruction are shown in Table 4–5.

If the result of the mantissa arithmetic is floating-point zero for the first result word for any of the first five of these instructions, the factors which affect the stored result are the content of D8 of the PSR, the magnitude of the characteristic produced for the word, and the sign of the input operand or operands for the instruction, as follows:

- If D8 = 0 (specifies normal operating mode for UNIVAC 1108),characteristic arithmetic has no effect and the result word is stored as +0.
- If D8 = 1 (specifies compatibility with UNIVAC 1107 floating-point zero conventions) and if the characteristic is in the range 000_8 through 377_8, the first result word reflects the magnitude of the characteristic produced, and also the sign produced either by the mantissa arithmetic or by consideration of the signs of the input operands.

INSTRUCTION	IDENTIFYING NAME OF RESULT WORDS	
	FIRST WORD	SECOND WORD
Floating Add	Sum	Residue
Floating Add Negative	Difference	Residue
Floating Multiply	Most Significant Word	Least Significant Word
Floating Divide	Quotient	Remainder
Load And Convert To Floating	Result	None (only one result word produced)
Floating Compress And Load	Result	

Table 4–5. Instructions Producing Results In Single-Precision Floating-Point Format

If D8 = 1 and the characteristic produced during the execution of the Floating Add, Floating Add Negative, or Load And Convert To Floating instruction is not within the range 000_8 through 377_8, the first word of the result is stored as +0 regardless of the signs of the input operands. For these instructions, the characteristic produced could never be greater than 377_8 if the mantissa produced is ±0.

If D8 = 1 and the characteristic produced during the execution of the Floating Multiply or Floating Divide instruction is not within the range 000_8 through 377_8, the first result word is stored as +0 or -0, whichever agrees with the sign of the input operands.

If the mantissa produced during the execution of a Floating Compress And Load instruction is either +0 or -0, the result stored is +0 regardless of the characteristic produced and the value of D8.

Characteristic arithmetic can produce a characteristic representing a number greater than 377_8 for the second result word only in the Floating Multiply instruction. Characteristic arithmetic can produce a characteristic representing a negative number for the second result word for any of the four arithmetic instructions which produce two-word single-precision floating-point format results. When an out-of-range characteristic is produced for the second result word, neither floating-point characteristic overflow or underflow is reported solely as the result of this occurrence.

If the characteristic calculated for the second result word in a Floating Multiply instruction represents a number greater than 377_8, the 9-bit value (or its ones complement, if the mantissa is negative) is packed with the mantissa. If overflow or underflow occurs on the second word of the product, it must have also occurred for the first word of the product. If the mantissa of the first word of the product is zero, however, the floating-point characteristic overflow fault interrupt does not occur and the CPU attempts to pack the second word of the product.

If the characteristic calculated for the second result word for any of the first four instructions listed in Table 4–5 represents a negative number, a +0 or -0 is stored, whichever reflects the sign of the word that would have been stored had the characteristic been in the range 000_8 through 377_8.

4.6.2. Double-Precision Floating-Point Zero

If the result of mantissa arithmetic is floating-point zero for any one of the six instructions which produces a result in double-precision floating-point format, neither floating-point characteristic overflow nor underflow will be reported even though the characteristic arithmetic may produce a characteristic which represents a number greater than 3777_8 or a negative number.

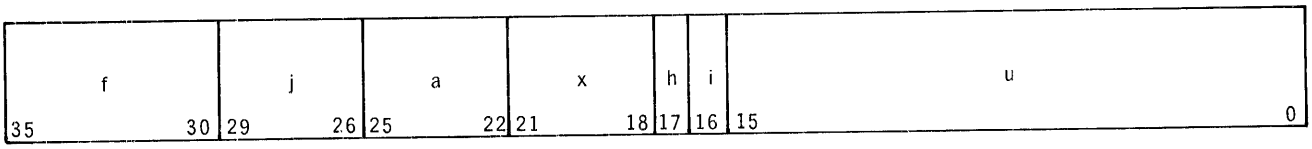
In all cases in which the mantissa produced for a double-precision floating-point result is floating-point zero, the result stored is +0, regardless of the signs of the input operands. The value of D5 in the PSR is ignored in these cases.

If the result of the mantissa arithmetic is not floating-point zero for any instruction which produces a result in double-precision floating-point format, but the characteristic for the result represents a negative number and D5 = 1, floating-point underflow is not reported. Instead the result is stored as +0. This combination of circumstances will never occur for the Floating Expand And Load instruction.

5. CPU CONTROL SECTION

5.1. INSTRUCTION WORD FORMAT

During the running of a program in the UNIVAC 1108 Central Processor Unit (CPU), instructions are transferred from locations in main storage to the control section of the CPU. The instructions are normally transferred from sequentially addressed main storage locations until the sequence is broken by the program or interrupted by the control section's reaction to some special condition or event. Each instruction is a coded directive to the control section; the control section initiates a sequence of steps necessary to perform the particular operation prescribed by the instruction. The 36-bit instruction word, illustrated below, is subdivided into seven fields.



where:

f = Function Code

j = Operand Qualifier, partial Control Register Address, or Minor Function Code

a = A, X, or R register; Channel, Jump Key, Stop Keys, or Module Number; Minor Function Code; partial Control Register Address

x = Index Register

h = Index Register Incrementation

i = Indirect Addressing

u = Operand Address or Operand Base

5.2. INSTRUCTION WORD FIELDS

The following paragraphs describe the manner in which the CPU's control section reacts to the contents of each of the seven fields of an instruction word. (U is the output of the index adder.)

5.2.1. Description of f Field

The f field is used to define the basic operation to be performed for all legal values of f less than or equal to 70_8. When the value in the f field is greater than 70_8, the f and j fields are combined to form a 10-bit field which defines the basic operation. For three of these f, j combinations, the value in the a field is used to define variations of the basic operation. All function codes are defined in Section 6 and listed in Appendix E.

5.2.2. Description of j Field

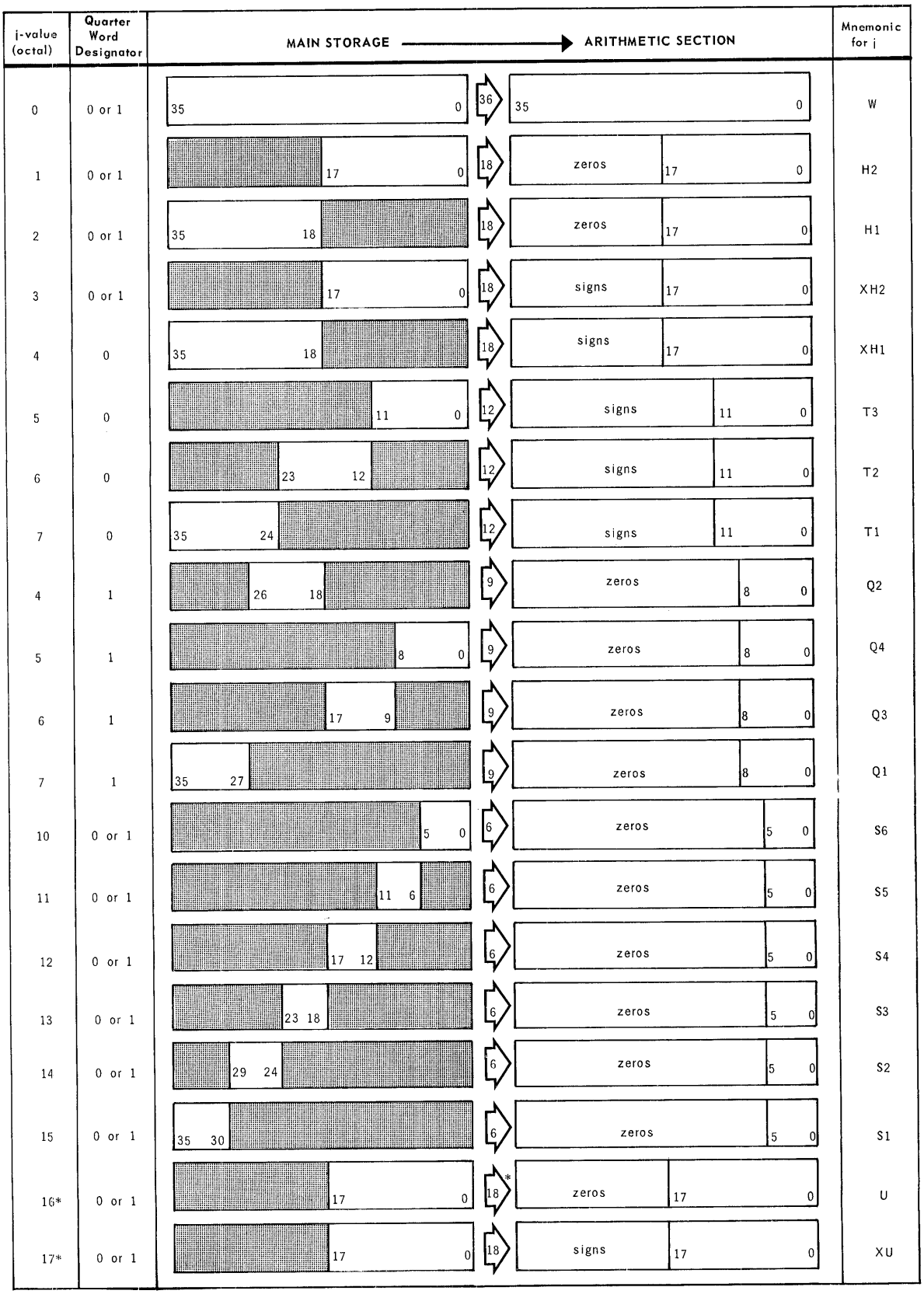
When f is less than 70_8, the j field is used as an operand qualifier. When f is equal to 70_8, the j field is used as part of a control register address. When f is greater than 70_8, the j field and the f field are used to define a basic operation, and, in this instance, the j field operates as a minor function code.

5.2.2.1. j Field As An Operand Qualifier

When the f field of an instruction contains a value in the range 01_8 through 67_8, the j field is used as an operand qualifier which specifies the data transfer pattern to or from main storage.

The j field can contain values ranging from 00_8 through 17_8. Each value except 4_8 through 7_8 determines a specific data transfer pattern. Each of the j field values 4_8 through 7_8 may specify either of two different data transfer patterns, with the choice dependent on the contents of the quarter word designator in bit 17 of the Processor State Register (PSR). (See 9.2.10.) Figures 5–1 and 5–2 illustrate all the possible data transfer patterns which can be specified by the j field.

The CPU included in Group Type 3011–99 does not have the quarter word Designator feature or quarter word transfer capability, and each j field value of 4_8 through 7_8 specifies only one data transfer pattern.



* See 5.2.8.2.

Figure 5–1. Transfers from Main Storage to the Arithmetic Section ($f = 10_8$ through 67_8)

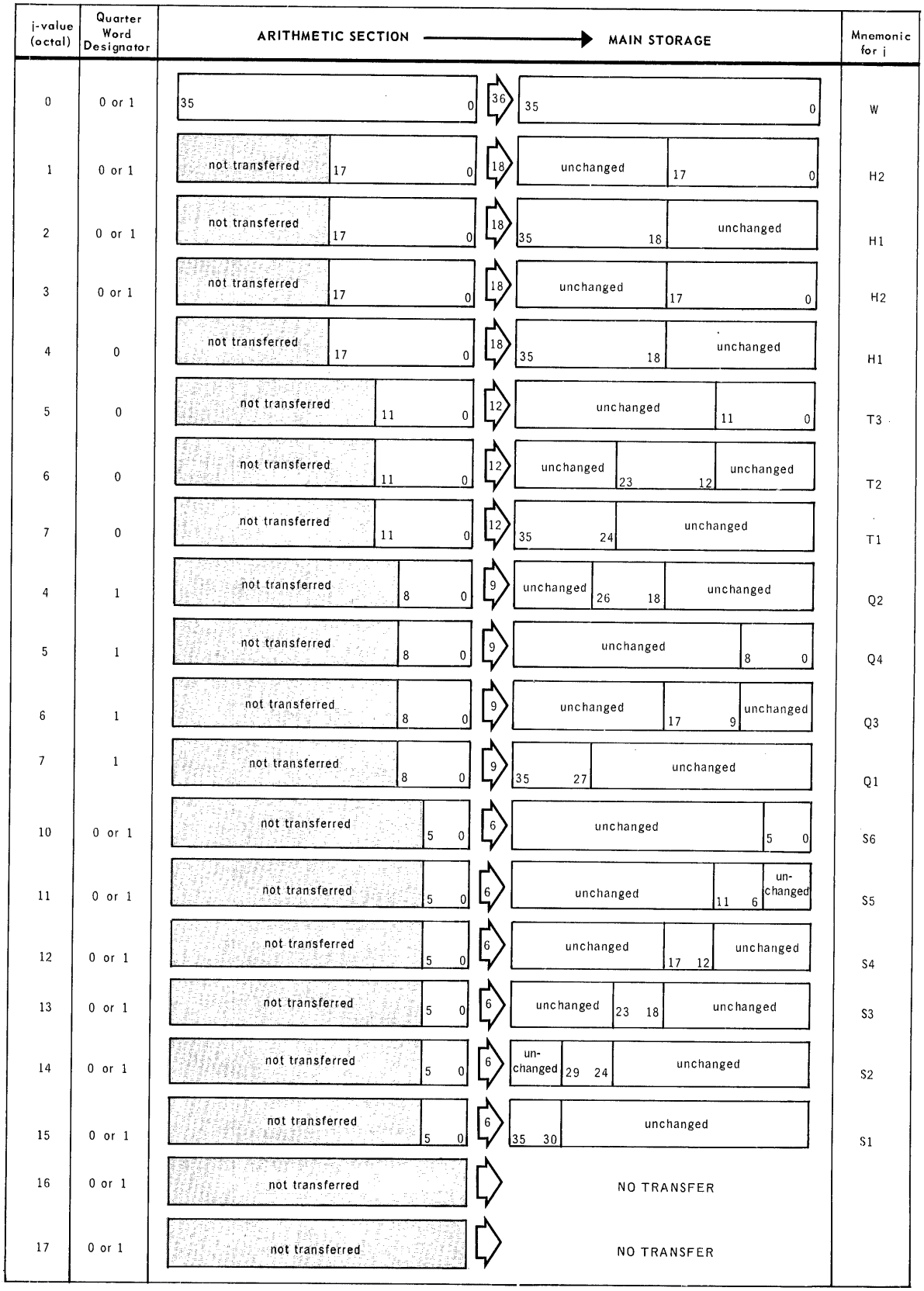


Figure 5–2. Transfers from the Arithmetic Section to Main Storage ($f = 01_8$ through 06_8 and 22_8)

5.2.2.1.1. Operand Qualification For Store and Block Transfer Instructions

The full 36-bit word in the control register specified by the a field (see 5.2.3.1) is transferred to a nonaddressable register in the arithmetic section (f = 01_8 through 04_8 and 06_8). The nonaddressable arithmetic register is cleared to +0 when f = 05_8.

- If j = 00_8, the full 36-bit word is transferred from the arithmetic section to the location (main storage or control register) specified by U.
- If j = 01_8 through 15_8 and U specifies a main storage location (U $\geq$ 200_8), a partial word is transferred from the least significant bit positions of the nonaddressable arithmetic register to specific bit positions (see Figure 5–2) of the main storage location. The contents of the remaining bit positions of the main storage location are not changed. Partial word writes, with lengths of third word, quarter word, or sixth word, increase the main storage cycle time to 1125 nanoseconds.
- If j = 01_8 through 15_8 and U specifies a control register (U $\leq$ 177_8), the j field is treated as if it contained 00_8, and the full 36-bit word is transferred to the control register.
- If j = 16_8 or 17_8, data is never transferred from the arithmetic section to any storage location (main storage or control register).

5.2.2.1.2. Operand Qualification When f = 10_8 through 67_8

These instructions require the transfer of a full 36-bit word or a partial word to the arithmetic section.

- If j = 00_8, the full 36-bit word addressed by U (see 5.3.3.1) is transferred to the arithmetic section.
- If j = 01_8 through 15_8 and U specifies a main storage location (U $\geq$ 200_8), a partial word is transferred to the arithmetic section. In the arithmetic section, the partial word is extended to a full 36-bit word either by zero fill or by sign bit fill from the leftmost bit position of the partial word, as illustrated in Figure 5–1.
- If j = 01_8 through 15_8 and U specifies a control register (U $\leq$ 177_8), the j field is treated as if it contained 00_8 and the full 36-bit word is transferred from the control register to the arithmetic section.
- If j = 16_8 or 17_8, an 18-bit partial word is transferred to the arithmetic section. Details on the formation of this partial word and its extension are given in 5.2.8.2.

5.2.2.2. Use of j Field as Partial Control Register Address

When f = 70_8, the most significant bit of the j field is ignored by the hardware, and the three low-order bits are combined with the contents of the a field to form a 7-bit control register address as described in 5.2.4.

5.2.2.3. Use of j Field as Minor Function Code

When f = 71_8 through 76_8, the value in the j field is a minor function code designator. An explanation of the details of each of these instructions is given in Section 6; they are summarized in Appendix E.

5.2.3. Description of a Field

The contents of the a field of an instruction word has a number of uses. The exact use is dependent on the instruction being performed and, in many cases, on the contents of the Processor State Register (PSR).

5.2.3.1. Use of the a Field to Reference A Register

For most of the instructions, the value in the a field references one of the A registers. When the control register selection designator (EXEC ABR), D6, of the PSR = 0, each value in the range 00_8 through 17_8 in the a field references one of the user A registers in the range of control register addresses 14_8 through 33_8, respectively. When D6 = 1, each value in the range 00_8 through 17_8 in the a field references one of the Executive A registers in the range of control register addresses 154_8 through 173_8 respectively. In some instructions, the value in the a field references two or three A registers. When two or three A registers are referenced, the value in the a field explicitly references register A_a, and implicitly references registers $A_a + 1$ and $A_a + 2$.

The unassigned control registers (addresses 34_8, 35_8, 174_8, and 175_8) can be used as extensions of the two sets of 16 A registers. For example, when a = 17_8 and the instruction requires the referencing of A registers (A_a and $A_a + 1$) then:

- If D6 = 0, the last user A register (address 33_8) is referenced for A_a, and the first user unassigned control register at address 34_8 is referenced for $A_a + 1$.
- If D6 = 1, the last Executive A register at address 173_8 is referenced for A_a, and the following Executive unassigned control register at address 174_8 is referenced for $A_a + 1$.

5.2.3.2. Use of the a Field to Reference X Registers

For certain instructions, the value in the a field references one of the X registers. When D6 = 0, each value in the range of 01_8 through 17_8 in the a field references one of the user X registers in the range of control register addresses 01_8 through 17_8 respectively; if a = 0_8, the PSR Temporary Storage Register at control register address 000_8 is referenced. The PSR Temporary Storage Register must not be used for storage of program information. When D6 = 1, each value in the range of 01_8 through 17_8 in the a field references one of the Executive X registers in the range of control register addresses 141_8 through 157_8 respectively; if a = 0_8, the Executive nonindexing X register at control register address 140_8 is referenced.

5.2.3.3. Use of the a Field to Reference R Register

For certain instructions, the value in the a field references one of the R registers. When D6 = 0, each value in the range of 00_8 through 17_8 in the a field references one of the user R registers at control register addresses 100_8 through 117_8, respectively. When D6 = 1, each value in the range of 00_8 through 17_8 in the a field references one of the Executive R registers at control register addresses 120_8 through 137_8, respectively.

5.2.3.4. Use of the a Field to Reference I/O Channels

For most input/output (I/O) instructions, each value in the range of 00_8 through 17_8 in the a field (inclusively OR'ed with the contents of the Channel Select Register) references one of the I/O channels 00 through 15 (00_8 through 17_8), respectively.

5.2.3.5. Use of the a Field to Reference Jump Keys

For a Jump On Keys instruction, each value in the range of 01_8 through 17_8 in the a field references one of the 15 selective jump keys on the operator's Display Console.

5.2.3.6. Use of the a Field to Reference Halt Keys

For a Halt On Keys And Jump instruction, each of the four bit positions in the a field references one of the four selective step keys on the operator's Display Console.

5.2.3.7. Use of the a Field to Modify Memory Select Register (MSR)

For the Select Interrupt Locations instruction, the value in the three low-order bits of the a field is transferred to the MSR.

5.2.3.8. Use of the a Field as Minor Function Code

For Store Channel Number, Initiate Interprocessor Interrupt/Alarm/Disable Day Clock/Enable Day Clock, or Load Channel Select Register/Load Last Address Register instruction, the value in the a field specifies a particular variation of the basic operation initiated by the f, j combination.

5.2.4. Use of the j and a Fields To Modify Control Register Address

For Jump On Greater And Decrement instruction, the values in the j field and a field combine to form a 7-bit address (the leftmost bit of the j field is ignored). The 7-bit address specifies which one of the 128 addressable control registers is to be used as the counter for the instruction.

5.2.5. Description of the x Field

An indexing operation which utilizes a ones complement subtractive adder occurs for every instruction. If the control register selection designator D6 of the PSR is equal to 0, each x field value in the range 01_8 through 17_8 references one of the user x registers at control register addresses 01_8 through 17_8, respectively. If D6 = 1, each x field value in the range 01_8 through 17_8 references one of the Executive X registers at control register addresses 141_8 through 157_8, respectively. When the value in the x field is not zero, the contents of the lower half (Xm) of the X register specified by the x field is added to the extended contents of the u field to form the modified operand address or a modified operand. This indexing operation is symbolized by the notation: u + Xm = U.

When the value in the x field is zero, no index register is referenced. However, an indexing operation does occur. It consists of adding an 18-bit half word of all 0 bits to the extended u field value to form the modified operand address or modified operand. This indexing operation is symbolized by the notation: u + 0 = U.

An indexing operation never produces a U value consisting of all 1 bits.

5.2.6. Description of the h Field

If the x field of the instruction word contains a nonzero value, the contents of the h field determines whether or not the contents of the X register specified by the x field are modified.

After the indexing operation is complete, if h = 1 and the x field is not zero, the contents of the upper half (Xi) of the addressed X register is added to the contents of the lower half (Xm) of the same X register and the sum is stored back in the lower half (Xm). The process is Xm + Xi → Xm. The addition is performed in an 18-bit ones complement subtractive adder in the index subsection. The modification of Xm is performed without increasing the instruction execution time.

The only time the index register modification process produces an output consisting of -0 is when both inputs to the process consist of -0, that is, (-0) + (-0) = -0.

If h = 0, the index register is not incremented or decremented.

In certain cases, the h field is used as an extension of the value in the u field (see 5.2.8.2).

5.2.7. Description of the i Field

The i field can be used to specify indirect or absolute addressing, or to extend the u field of an instruction.

If i = 1 and D7 = 0, indirect addressing occurs for all instructions except when f = 01_8 – 67_8 and j = 16_8 or 17_8. For the exception, x ≠ 0 is also a required condition for indirect addressing.

Indirect addressing will not occur if:

i = 0, or

f = 01_8 – 67_8, j = 16_8 or 17_8, x = 0; the i field is used as an extension of the u field.

When D7 = 1, indirect addressing is not possible and i = 1 specifies absolute addressing. The i field may still be used as an extension of the u field.

The exception cases are summarized in Table 5–1.

CONTENTS OF i FIELD	CONTENTS OF BASE REGISTER SUPPRESSION BIT (D7) OF PSR	IF f = 01_8 THRU 67_8; j = 16_8 OR 17_8; AND	
		x FIELD ≠ 0	x FIELD = 0
0	0	Normal Addressing	For all values of i field and D7, the i field plus the h field is used to extend the u field
	1		
1	0	Indirect Addressing	
	1	Absolute Addressing	

Table 5–1. Use of i Field

Indirect addressing is initiated after calculating the relative address and completing the conversion to absolute address in the index subsection, even if U ≤ 177_8. The contents of bit positions 21 through 0 of the main storage location addressed are transferred to the control section of the CPU, where they replace the x, h, i and u field values of the current instruction. The modified instruction is then performed just as if the whole instruction word were initially obtained in its modified form from main storage. Indexing and index register incrementation (if specified) are performed in the normal manner for both the original and the modified instruction. If the modified instruction also specifies indirect addressing, the whole process of indirect addressing is repeated. The repetition or cascading of indirect addressing continues until the modified instruction contains a 0 bit in the i field, or contains all 0 bits in the x field for certain f, j combinations listed in Table 5–1, at which time indirect addressing ceases and the balance of the instruction is performed.

If f = 01_8 through 67_8, j = 16_8 or 17_8, and x = 0 in an instruction as it is initially obtained from main storage or as it is modified as a result of an indirect addressing operation, indirect addressing does not occur even if i = 1. In this case, the i field is used as an extension of the u field.

5.2.8. Description of the u Field

The ultimate use of the u field depends on the values in the f and j fields of the instruction.

For most f,j combinations, u is used as an operand address designator. The value in the u field after indexing is used as the relative address of a main storage location or as a control register address.

For certain f,j combinations, the indexed extension of the value in the u field of the instruction (or of a modified instruction in the case of indirect addressing) is used as the operand for some instructions or as a count in the case of shift instructions. For other f,j combinations, the value in the u field has no effect on the result of the instruction.

5.2.8.1. Use of the u Field as an Operand Address Designator

The value in the u field of an instruction (or a modified instruction resulting from an indirect addressing sequence) is an operand address designator if:

- f = 01_8 through 67_8 and j = 00_8 through 15_8;
- f = 70_8 through 72_8, 75_8, or 76_8; or
- f = 73_8 and j = 06_8, 07_8, or 17_8.

It is also an operand address designator if indirect addressing is being performed for any instruction.

When the value in the u field is an operand address designator, the 16-bit value in the u field is always extended to an 18-bit input to the index adder by appending 0 bits to form the two leftmost bit positions. The 18-bit output of the index adder (U) is used as a relative main storage address or as a control register address. When U < 200_8, U is used as a control register address if it is not being used as an indirect address, a jump-to address, or the (remote) address for an Execute instruction. U is used as a relative main storage address in all other cases.

For any given u field value, a value can be chosen for the Xm portion of the index register specified by the x field which will produce any desired value of U in the range 000000_8 through 777776_8. (It is not possible to produce the value 777777_8.)

Certain instructions use U to reference both U and U + 1 as a double-length (72-bit) word. In this case, U is the address of the most significant 36 bits and U + 1 is the address of the least significant 36 bits.

5.2.8.2. Use of the u Field as an Operand Designator

The value in the u field of an instruction (or a modified instruction) is an operand designator if indirect addressing is not specified and

- f = 10_8 through 67_8 and j = 16_8 or 17_8; or
- f = 73_8 and j = 00_8 through 05_8 or 10_8 through 13_8 (all shift instructions)

When the value in the u field of an instruction (or a modified instruction resulting from an indirect addressing sequence) is an operand designator, the 16-bit value in the u field is extended to 18 bits to provide one of the inputs to the index adder for an indexing operation. This 18-bit value normally consists of 0 bits in the two leftmost bit positions and the 16-bit value from the u field in the remaining bit positions. However, if f = 10_8 through 67_8, j = 16_8 or 17_8, and x = 0, the bits in the h and i fields are used in the two leftmost bit positions in place of the 0 bits. When h and i are both 1 bits and they are used to extend a u field whose value is all 1 bits, the 18-bit output of the index adder is all 0 bits rather than all 1 bits.

The 18-bit index adder output is normally sent to the arithmetic section where it is extended to become a 36-bit operand by 0-bit fill (j = 16_8) or by filling with bits identical to the leftmost bit of the index adder output (j = 17_8).

5.2.8.3. Restrictions of Use of the u Field

The u field is not used in the following instructions when their j field contains 16_8 or 17_8 and their i field contains a 0 bit.

- Store A
- Store Negative A
- Store Magnitude A
- Store R
- Store Zero
- Store X

In the following instructions, the u field is not used when their i field contains a 0 bit.

- No Operation
- Disconnect Input Channel
- Disconnect Output Channel
- Allow All Channel External Interrupts
- Prevent All Channel External Interrupts
- Executive Return
- Initiate Interprocessor Interrupt
- Select Interrupt Locations

5.3. GENERAL OPERATION OF THE CONTROL SECTION

The CPU's control section automatically obtains instruction words from main storage, decodes them, and provides the control signals needed to execute the instruction. The control section contains the following:

- Program address counter (P register)
- Program control subsection
- Index subsection
- Storage class control subsection

5.3.1. Program Address Counter (P Register)

The instruction words of the program are stored in one or more groups of consecutive main storage locations. When a program is executed, each instruction word is transferred from its main storage location to the control section. In the control section, the instruction word is decoded and the control signals needed to execute the instruction are generated. The main storage location of the instruction word is specified by the contents of the P register. The time at which an instruction word is transferred to the control section is controlled by the program control subsection.

The 18-bit P register contains the absolute address of the next instruction to be executed. Its contents are automatically incremented by one during the early stages of the instruction's execution. The P register contents are again incremented by one during the closing stages of the execution of a Search, Masked Search, or Test instruction for which a skip next instruction condition is met. The P register is not incremented following the 22-bit main storage to control section transfer for an indirect addressing sequence or following the transfer of a word addressed by U during the execution of an Execute instruction.

The automatic addressing of sequential main storage location is altered by the execution of a Jump instruction. When a Jump instruction is performed, the address in the P register is replaced by an absolute main storage address derived in the index subsection. The next instruction word is obtained from the main storage location specified by the altered contents of the P register.

These results are a consequence of the nature of incrementation, which is a true addition rather than addition by complement subtraction. For example:

- If the P register contains 777776_8, incrementation produces the value 777777_8 rather than 000000_8.
- If the P register contains 777777_8, incrementation produces the value 000000_8 rather than 000001_8.

5.3.2. Program Control Subsection

The program control subsection contains a number of registers, including the F0, F1, F3, and F4 registers.

Each instruction word is transferred from the main storage location to the F0 register at the appropriate time. While in the F0 register, the instruction word is interpreted so as to provide the control signals for the operations of indexing, index register incrementation, and indirect addressing. The controls for the reading of input operands for the instruction and the storing of results are also set up while the instruction is in the F0 register. While these operations are being performed, values are transferred to the F1, F3, and F4 registers as follows:

- An image of the f field and j field values and the logical sum (inclusive OR) of the a field value and the contents of the Channel Select Register (CSR) are transferred to the F1 register. The f field and j field values in the F1 register are interpreted to provide control signals in the CPU's arithmetic section for activities such as arithmetic computations, logical operations, compares, tests, shifts, and partial word transfers. When an I/O instruction is being performed, the logical sum of the contents of the a field and the F1 register (previous contents of CSR) specify an I/O channel and the location of an input or output Access Control Word (ACW) register.

- The a field value in the F0 register is modified (the modification depends on the f field value and the condition of the control register selection designator D6) and transferred to the F3 register. The contents of the F3 register now specifies the 7-bit address of a control register for instructions which use the a field to specify an A, X, or R register. The 7-bit address of control register $A_a + 1$ is transferred to the F4 register for use by instructions which require it. The addresses in the F3 and F4 registers are used to obtain input operands and to store results for many instructions. When needed, the 7-bit address of control register $A_a + 2$ is derived by from the contents of the F4 register. The addresses in the F3 and F4 registers are also used to detect certain anomalies and conflicts as described in 5.4.1 and 5.4.3.

5.3.3. Index Subsection

The arithmetic operations of indexing, index modification, repeat counter decrementation, real time clock decrementation, and input/output Access Control Word modification are performed in the index subsection using an 18-bit ones complement subtractive index adder. The 18-bit index adder produces an output of all 1 bits only when both inputs consist of all 1 bits.

5.3.3.1. Indexing

When the x field of the current instruction in the F0 register is not zero, the value Xm, from the lower half of the X register specified by the x field, is added to the extended contents of the u field. If the x field is zero, +0 is added to the extended contents of the u field. The contents of the 16-bit u field are normally extended to 18 bits by supplying 0 bits as the leftmost two bits. If the f field contains a value in the range 01_8 through 67_8, and if x = 0 and j = 16_8 or 17_8, then the bits from the h and i fields of the instruction are used to extend the u field to 18 bits. The result of the indexing operation is an 18-bit value called U. The use of U is determined in the storage class control subsection.

The indexing operation uses an 18-bit ones complement subtractive adder having the same general characteristics as the main adder described in 4.2. It should be noted that when U is an operand address and the CPU is operating in the 1107 system compatibility mode (D4 of the PSR = 1), 0 bits are always used as the two leftmost bits of U, regardless of the index adder output.

Conversion of relative address to absolute address is also performed in the indexing subsection. (Complete details of this conversion are presented in 9.3). Briefly, a program's instruction words and data words can be stored in two portions of main storage; one portion is referred to as the I-blank and the other portion is referred to as the D-bank; each portion may contain instruction words or data words or both.

Conversion of relative address to absolute address involves two compound indexing operations, each using a 9-bit modified ones complement subtractive adder and an 18-bit ones complement subtractive adder. The 9-bit adders are said to be modified ones complement subtractive adders because an end-around borrow is always forced* from the leftmost bit position of the first phase result to obtain the second phase result. In all other respects, the 9-bit adders and the 18-bit adders used in the compound indexing operations follow the rules specified in the explanation of the main adder in 4.2.

The minuend input for one of the 9-bit adders is the 9-bit BI field of the PSR; the minuend input for the other is the 9-bit BD field of the PSR (see 9.3.6 for exceptions). The values of BI and BD indicate the displacement of the absolute addresses of the words in the I-bank and the D-bank, respectively, from the relative addresses of these words. The relative addresses of these words are assigned in the initial program assembly or compilation. The displacement is represented in terms of blocks of 512 words.

The subtrahend for both 9-bit adders is the 9-bit value consisting of 1 bits in the two leftmost bit positions and the ones complement of the contents of bit positions 15 through 9 of the u field.

The output of each 9-bit adder is extended to 18 bits by inserting the contents of the nine rightmost bit positions of the u field into the nine rightmost bit positions. The resultant BI + u and BD + u are each used as the minuend input to the two 18-bit adders involved in the compound additions. The subtrahend is the ones complement of Xm if the x field of the instruction is not zero, or all 1 bits if the x field is zero. The resulting compound sums are SI = (BI + u) + (Xm or 0) and SD = (BD + u) + (Xm or 0). They are absolute addresses. When the CPU is operating in the 1107 system compatibility mode (D4 of the PSR = 1), 0 bits are used as the two leftmost bits of the 18-bit sums SI and SD, regardless of the bits produced for these two bit positions.

The end-around borrow is forced, which permits developing a 9-bit output consisting of all 1 bits. Because of the method of forming the input from the u field, a 9-bit output of all 1 bits could not be produced by a true ones complement subtractive adder.

The sums U, SI, and SD are produced in parallel. Interpretation of the current instruction word and U in the storage class control subsection determines whether or not an operand location in main storage is addressed, and if addressed, controls the selection of either SI or SD to address main storage.

Conversion of relative to absolute address is also applicable to the address of the rightmost half of a double length operand. After U, SI, and SD have been produced, the values U+1, SI+1, and SD+1 are calculated.

It should be noted that addition of 1 to certain values produces the following results:

$$777775_8 + 1 = 777776_8$$
$$777776_8 + 1 = 000000_8$$
$$777777_8 + 1 = 000001_8$$
$$000000_8 + 1 = 000001_8$$

It is not possible to produce 777777_8 as the value for U+1, SI+1, or SD+1. Unless otherwise specified, mention of main storage addresses will hereafter be in terms of U (the relative address) in order to simplify the presentation.

5.3.3.2. Index Modification

If the value in the x field of the current instruction word is not zero and the h field contains a 1 bit, index modification occurs after the indexing operation is complete. During index modification, Xi (upper half of the X register specified by the x field) is added to Xm (the lower half of the same X register). The result of the addition of Xm and Xi is stored in the lower half of the index register.

5.3.3.3. Real Time Clock Decrementation

The lower half of the Real Time Clock register (R0 register) is decremented by 1 once every 200 microseconds, independently of program control.

5.3.3.4. Repeat Counter Decrementation

The lower half of the Repeat Count register (R1 register) is decremented by 1 during each pass through a repeated instruction. The decremented value replaces the current value in the R1 register during the termination pass for the instruction.

5.3.3.5. Input/Output Access Control Word Modification

In an I/O operation, an 18-bit field (V field) of the I/O Access Control Word specifies an absolute address in main storage to or from which an I/O data word is transferred. The value in another field (G field) of an Internally Specified Index I/O Access Control Word, or the value in the G and H fields of an Externally Specified Index I/O Access Control Word, specifies whether the value in the V field is to be incremented by 1, decremented by 1, or left unchanged each time an I/O data word is transferred to or from main storage.

The value in the W field of the I/O Access Control Word specifies the number of I/O data words to be transferred to or from main storage. The value in the W field is always decremented by 1 as each I/O data word is transferred to or from main storage.

The associated I/O operation is complete when the value in the W field has been decremented to zero. Additional information on I/O Access Control Words is contained in 7.2 and 7.3.

5.3.4. Storage Class Control Subsection

The values U, SI, and SD formed in the index subsection, are made available to the storage class control subsection for selection according to the instruction being performed. U may be used as the address of a control register, or as an operand within the arithmetic section. SI or SD are used to address main storage, or for a new value to be sent to the P register, but never to address control registers nor as an operand of the instruction.

If an address is required, selection is based on comparisons of U with BS of the PSR, and of U with 200_8. When U is compared with BS, it is actually a comparison of the leftmost nine bits of U with the 9-bit value produced by extending the 7-bit BS with two high-order 0 bits. In subsequent explanations, the result of this comparison is denoted by: $U \leq BS$ and $U > BS$.

- If an operand or shift count is required rather than an address, the 18-bit value U is sent to the arithmetic section.

- The address of an operand is selected according to the following rules:

  – If $U < 200_8$, U is transferred to the control register addressing circuitry.
  – If $U \geq 200_8$, and

    (1) if $U \leq BS$, SI addresses main storage; or

    (2) if $U > BS$, SD addresses main storage.

  When a two-word operand is required, the selection occurs twice. The first pass selects U, SI, or SD. During the second pass the decision is made regarding U+1, SI+1, and SD+1. The two consecutive addresses involved are not always consecutive absolute addresses. Special cases which illustrate this fact are:

  – If $U = 177_8$, a control register ($U+1 = 200_8$) and SI+1 are used to address main storage.

  – If U=BS, SI is the address of the first word; but if U+1 is greater than BS, SD+1 is the address of the second word.

- For a jump instruction, the new value for the P register is selected according to the rule:

  – If $U \leq BS$, SI is transferred to the P register.

  – If $U > BS$, SD is transferred to the P register.

  During each jump instruction, a designator within the CPU is set to indicate which of the two values was used to load the P register. The information is required whenever the absolute value in the P register must be reduced to a relative value by subtracting the contents of the appropriate base register.

- An indirect address or the address referenced in the Execute instruction is:

  – SI, if $U \leq BS$; or

  – SD, if $U > BS$.

5.4. CONTROL SECTION TIMING

The control section utilizes various timing chains to initiate the reading and writing of main storage locations and control registers as required for the execution of each instruction. Most chains control more than one activity, issuing the different commands necessary for this control at intervals throughout chain operation.

Progress along each timing chain is governed by a clock which defines 125 nanosecond cycles for the chain.

5.4.1. Basic Timing Chains

The following discussion of timing chains is intended to provide a basis for understanding the anomalies and control register conflicts described in 5.4.3. Examples of specific timing sequences for typical instructions are given in 5.4.2.

5.4.1.1. Main Timing Chain (T0 Chain)

The main timing chain (T0 chain) controls initiation of other timing chains which, in turn, control specialized activity within the control section, arithmetic section, I/O section, and main storage modules of the system. In some cases, the T0 chain must halt and wait for completion of part or all of the activity being performed under the control of one or more of the other timing chains. For instance, a main storage timing chain or an arithmetic timing chain for an extended sequence instruction may not have proceeded far enough to permit proper initiation of the next step of the T0 chain. The duration of the wait is a multiple of 125 nanoseconds. The T0 chain normally utilizes five productive cycles and one or more wait cycles, as follows:

- CYCLE 1 – This cycle is used to address the control register specified by the x field of the immediately preceding instruction in conjunction with D6 of the PSR. The control register is read (Read Xx sequence), and its contents (both Xm and Xi – the modifier and the increment portions, respectively) are transferred to the index subsection. The contents of the u field of the instruction, together with the contents of the h and i fields if $f < 70_8$, $j = 16_8$ or 17_8, and x=0, are also transferred to the index subsection.

- CYCLE 2 – During this cycle, the incremented contents of the P register are stored in that register: (P) + 1 → P.

  The U, SI, and SD values are produced in the index subsection. The storage class control subsection determines which of the three is to be used and how to use it. If the chosen value is to be used as the main storage address of an operand word to be sent to the arithmetic section, a result word to be stored, or a word from which the rightmost 22 bits are to be transferred to the F0 register for an indirect addressing sequence, the T0 chain initiates a request for a main storage read or write cycle (Request Read SI/SD sequence or Request Write SI/SD sequence).

  In processing those instructions which use the u field as an index adder input to define the addresses of the two words of a double precision operand or result, if the first word of the input operand or result has been read or stored in the preceeding pass through the T0 chain, this cycle is used to produce U+1, SI+1, and SD+1, and to determine which of the three values to use and how to use it. If a main storage reference is required, the T0 chain requests it (Request Read SI+1/SD+1 or Request Write SI+1/SD+1).

Cycle 2 is also used to request a main storage cycle to read the next instruction (NI), which is addressed by the value in the P register (Request Read NI), unless inhibited by one of the following conditions:

- The cycle is being used to initiate a main storage reference as part of an indirect addressing sequence.

- The nature of the instruction prevents clearing the F0 register in time to ensure that the next instruction can be accepted. This occurs when the first access is requested for an instruction which can require two main storage accesses for a double-precision input operand or result, or at any time during the execution of a Block Transfer, Search, or Masked Search instruction.

- A main storage conflict exists because the P register addresses the same main storage module as a Request Read/Write U/SI/SD sequence or a Request Read/Write U+1/SI+1/SD+1 sequence initiated during this cycle.

- The instruction is a Store instruction with j = 16_8 or 17_8, and the value U calculated in the index subsection would address the same main storage module as the value in the P register. The next instruction is not read in even though U is not actually used to address main storage.

■ CYCLE 3 – For most instructions, a control register is selected using an address derived from the contents of the a field of the instruction in conjunction with D6 of the PSR. The control register is read (Read Aa, Xa, Ra, Aa+1, or ja) and its contents are transferred to the arithmetic section.

At the start of this cycle, the appropriate chain is activated in the arithmetic section.

■ CYCLE 4 – When index incrementation is specified (h=1), cycle 4 is used to store Xm + Xi from the index subsection in bits 17 through 0 of the index register indicated by the contents of the x field in conjunction with D6 of the PSR (Write Xx).

For some instructions, this cycle is also used to select a control register using an address derived from the contents of the a field in conjunction with D6 of the PSR, to read the selected control register, and to transfer its contents to the arithmetic section (Read Aa or Aa+1).

If a Request Write U/SI/SD or Request Write U+1/SI+1/SD+1 sequence has been initiated in cycle 2, the data to be written is made available to the selected main storage module.

■ WAIT CYCLES – If one or two main storage cycles have been requested in cycle 2, there is a delay of 125 nanoseconds or an integral multiple thereof before cycle 5 of the T0 chain is initiated. Cycle 5 is not initiated until each main storage module requested to initiate a storage reference responds to that request. For a Request Write, the main storage module response indicates that the data made available in cycle 4 has been accepted. For a Request Read, the main storage response indicates that the requested data or instruction has been read and is now available to the CPU.

- CYCLE 5 – If a Request Read has been initiated in cycle 2, the data from main storage is sent to the arithmetic section or the F0 register as appropriate. The notation used in the examples of instruction execution sequences which follow is:

  (SI/SD) → Arithmetic Section

  (SI+1/SD+1) → Arithmetic Section

  NI → F0 register

  If the value U (or U+1) has been chosen in cycle 2 to address a control register rather than main storage, the selected control register is read and the data transferred to the arithmetic sections, or data from the arithmetic section is written in the selected control register.

Following the completion of cycle 5, the T0 chain is immediately restarted at cycle 1 unless an arithmetic timing chain initiated in cycle 3 has not yet signalled that the T0 chain may proceed. The events occurring during the following pass through the T0 chain vary, however. They depend on the nature of the instruction being performed, the progress already made in performing the instruction, and whether or not the next instruction has been transferred to the F0 register during cycle 5.

- If the instruction is an Execute instruction, the T0 chain is immediately restarted for the purpose of translating the subject instruction.

- If the low-order 22 bit positions of the F0 register have been loaded as the result of an indirect addressing sequence, the T0 chain is immediately restarted in order to develop U, SI, and SD.

- If not all of the storage references using addresses to be developed in the index subsection have been completed for an instruction requiring such multiple references (for example, a Double Precision Floating, Search, Masked Search, or Block Transfer instruction), the T0 chain is immediately restarted to initiate development of the next operand word or result word address.

- If all of the passes through the T0 chain needed for storage access are complete, the T0 chain is restarted for the purpose of translating the next instruction as soon a signal to proceed has arrived from any arithmetic timing chain initiated in cycle 3. Any delay while waiting for the signal is an integral multiple of 125 nanoseconds.

  If the next instruction has not been read from main storage during a preceding pass through the T0 chain because of a conflict in storage access, there is an additional pass through the T0 chain for the sole purpose of obtaining the next instruction. In this case, the only productive cycles of the chain are cycle 2 (Request Read NI) and cycle 5 (NI → F0 register).

5.4.1.2. Result Storage Chain (T1 Chain)

When the T0-chain is restarted to initiate the processes required by the next instruction or to obtain the next instruction from main storage, the Result Storage Chain (T1 chain) is also started, if necessary. The T1 chain sequences the storing of the result word or words from the arithmetic section in control registers. It utilizes a maximum of three productive cycles. These cycles coincide in time with cycles 1 through 3 of the T0 chain, which is operating in parallel with the T1 chain. The three cycles of the T1 chain are utilized as follows:

- CYCLE 1 – For the Double Load Shift And Count instruction, a value from the arithmetic section is stored in the control register addressed by the contents + 1 of the F4 register (Write Aa+2).

- CYCLE 2 – For all but two of the instructions, which require storing results in two control registers, this cycle is used to store a value from the arithmetic section in the control register addressed by the contents of the F4 register (Write Aa+1).

  For the Double Load And Unpack Floating and Double Load And Convert To Floating instructions, this cycle is used to store a value from the arithmetic section in the control register addressed by the contents + 1 of the F4 register (Write Aa+2).

  For the Load X Modifier, Load X Increment, and Load Modifier And Jump instructions, this cycle is used to store an 18-bit value from the arithmetic section in the index register addressed by the contents of the F3 register (Write Xa).

- CYCLE 3 – For most instructions (except Load X Modifier, Load X Increment, and Load Modifier And Jump) which require storing results in one or more control registers, this cycle is used to store a value from the arithmetic section in the control register addressed by the contents of the F3 register (Write Aa, Xa, Ra, ja, OACR, IACR, or Aa+1).

5.4.2. Alternate Bank vs. Same Bank Timing

The time required to perform an instruction is defined as the time from the start of cycle 1 of the T0 chain which follows the loading of the instruction in the F0 register, until the start of cycle 1 of the T0 chain which follows the loading of the next instruction in the F0 register. The time required to access the next instruction and load it in the F0 register overlaps the execution of the current instruction.

An instruction is said to be performed with alternate bank timing if the same main storage module is *not specified* by the address of the next instruction and the last value selected in the storage class control subsection for the current instruction (U, SI, or SD for single pass instructions; U + 1, SI + 1, or SD + 1 for two-pass instructions). Alternate bank timing applies (main storage module conflict is not detected) if the selected value is an input operand or shift count, if it addresses a control register, or if it addresses a main storage module other than the one containing the next instruction. In this case, activities for the current instruction completely overlap the accessing of the next instruction. Addressing a main storage module for the next instruction can delay completion of the current instruction by 125 nanoseconds or a multiple thereof if access to the module is delayed by a wait for completion of a main storage cycle initiated for one of the other processors in the system or by a wait for initiation and completion of a main storage cycle for a higher priority processor.

An instruction is said to be performed with same bank timing if the same main storage module is specified by the last value selected in the storage class control subsection for the current instruction and the address of the next instruction. Same bank timing applies only if the selected value is an address and it specifies the main storage module containing the next instruction. In this case, the current instruction and the next instruction can overlap only during cycles 2 and 3 of the T1 chain for the current instruction.

The following paragraphs explain in detail the steps performed in the various passes through the T0 chain and the T1 chain for a single pass instruction and for a two-pass instruction.

5.4.2.1. Typical Single Pass Instruction Timing

The timing sequences used when a single-precision fixed-point add instruction is performed are representative of the sequences used for many instructions. The following listings show pertinent steps in the timing sequences for both the alternate bank case and the same bank case when performing the Add To A instruction. The alternate bank sequence applies if $U \leq 177_8$; if $j = 16_8$ or 17_8; or if $U \geq 200_8$, $j \neq 16_8$ or 17_8, and the value in the P register addresses a word (the next instruction) in a different main storage module than the word addressed by SI or SD. The same bank sequence applies only if $U \geq 200_8$, $j \neq 16_8$ or 17_8, and the value in the P register addresses a word in the same main storage module as the word addressed by SI/SD. In examples 1 and 2, the listed steps include the reading of the Add To A instruction from main storage, the pass through T0 chain to supply input operands to the arithmetic section, the work performed in the T1 chain to store the sum produced by the arithmetic section, and the reading of the next instruction from main storage.

EXAMPLE 1:

ALTERNATE BANK TIMING

Add To A Instruction

PRIOR INSTRUCTION	ADD TO A	NEXT INSTRUCTION
T0 CYCLE 1	·	
T0–2	T0–2 Request Read NI	
T0–3	·	
T0–4	·	
T0–5	T0–5 NI→F0 register	
T1 CYCLE 1	T0 CYCLE 1 Read Xx; u→Index Subsection	·
T1–2	T0–2 (P)+1→P; Request Read SI/SD	T0–2 Request Read NI (no main storage module
T1–3	T0–3 Read Aa; Start Arithmetic Sequence	· conflict between SI/SD and P)
	T0–4 Write Xx	·
	T0–5 (SI/SD)→Arithmetic Section	T0–5 NI→F0 register
	T1 CYCLE 1	T0 CYCLE 1
	T1–2	T0–2
	T1–3 Write Aa (result of add instruction)	T0–3
		T0–4
		T0–5

EXAMPLE 2:

SAME BANK TIMING

Add To A Instruction

PRIOR INSTRUCTION	ADD TO A	EXTRA INSTRUCTION FETCH SEQUENCE AND NEXT INSTRUCTION
T0 CYCLE 1	.	
T0–2	T0 CYCLE 2: Request Read NI	
T0–3	.	
T0–4	.	
T0–5	T0–5 NI→F0 Register	
T1 CYCLE 1	T0 CYCLE 1: Read Xx; u→Index Subsection	.
T1–2	T0–2 (P)+1→P; Request Read SI/SD	T0–2 (Cannot initiate Request Read NI
T1–3	T0–3 Read Aa; Start Arithmetic Sequence	. because of main storage module
	T0–4 Write Xx	. conflict between SI/SD and P.)
	T0–5 (SI/SD)→Arithmetic Section	.
	T1 CYCLE 1	T0 CYCLE 1
	T1–2	T0–2 Request Read NI
	T1–3 Write Aa (Result of add instruction)	T0–3
		T0–4
		T0–5 NI→F0 Register
		TO CYCLE 1
		T0–2
		.
		.
		.

5.4.2.2. Typical Two-Pass Instruction Timing

The Double Precision Fixed Point Add instruction uses two passes through the T0 chain to obtain input operands for the arithmetic section. Its timing sequences are representative of the sequences used for all instructions which require input to the arithmetic section of two words whose addresses are derived in the index subsection using the u field of the instruction. The following listings show pertinent steps in the timing sequences for both alternate bank and same bank timing. The alternate bank sequence applies if U+1 $\leq 177_8$, or if U+1 $\geq 200_8$ and the value in the P register addresses an instruction which is not in the same main storage module as the operand word addressed by SI/SD+1. The same bank sequence applies only if U+1 $\geq 200_8$ and the value in the P register addresses a word in the same main storage module as the word addressed by SI+1/SD+1. In examples 3 and 4, the listed steps include the reading of the instruction from main storage, two passes through the T0 chain to supply input operands to the arithmetic section, the work performed by the T1 chain to store the sum produced in the arithmetic section, and the reading of the next instruction from main storage.

EXAMPLE 3:

ALTERNATE BANK TIMING

Double Precision Fixed Point Add Instruction

PRIOR INSTRUCTION	DP ADD TO A	NEXT INSTRUCTION
T0 CYCLE 1	·	
T0–2	T0–2 Request Read NI (the DA Instruction)	
T0–3	·	
T0–4	·	
T0–5	T0–5 NI→F0 Register	
T1 CYCLE 1	T0 CYCLE 1 Read Xx; u→Index Subsection	
T1–2	T0–2 (P)+1→P Request Read SI/SD.	
T1–3	T0–3 Read Aa; Start Arithmetic Sequence	
	T0–4 Write Xx	
	T0–5 (SI/SD)→Arithmetic Section	
	T0 CYCLE 1	·
	T0–2 Form U+1, SI+1 and SD+1 in the Index Subsection; Request Read SI+1/SD+1.	T0–2 Request Read NI (no main storage conflict between
	T0–3 Read Aa+1	· SI+1/SD+1 and P)
	T0–4	·
	T0–5 (SI+1/SD+1)→Arithmetic Section	T0–5 NI→F0 Register
	T1 CYCLE 1	T0 CYCLE 1
	T1–2 Write Aa+1	T0–2 etc.
	T1–3 Write Aa	· ·

EXAMPLE 4:

SAME BANK TIMING

Double Precision Fixed Point Add Instruction

PRIOR INSTRUCTION	DP ADD TO A	NEXT INSTRUCTION
T0 CYCLE 1	·	
T0–2	T0–2 Request Read NI (the DA instruction)	
T0–3	·	
T0–4	·	
T0–5	T0–5 NI→F0 Register	
T1 CYCLE 1	T0 CYCLE 1 Read Xx; u→Index Subsection	
T1–2	T0–2 (P)+1→P; Request Read SI/SD	
T1–3	T0–3 Read Aa; Start Arithmetic Section	
	T0–4 Write Xx	
	T0–5 (SI/SD)→Arithmetic Section	
	T0 CYCLE 1	·
	T0–2 Form U+1, SI+1, & SD+1 in Index Subsection; Request Read SI+1/SD±1.	T0 CYCLE 2 (Cannot initiate Request Read NI because
	T0–3 Read Aa+1	· of main storage access conflict
	T0–4	· between SI+1/SD+1 and P.)
	T0–5 (SI+1/SD+1)→Arithmetic Section	·
	T1 CYCLE 1	T0 CYCLE 1
	T1–2 Write Aa+1	T0–2 Request Read NI
	T1–3 Write Aa	T0–3
		T0–4
		T0–5 NI→F0 register
		T0 CYCLE 1
		T0–2 etc.
		·

5.4.3. Anomalies and Conflicts

The circuitry associated with the writing into and reading from the set of 128 control registers is designed to provide reliable operation for simultaneously writing in one control register and reading from another. Reliability may be impaired, however, if an attempt is made to write in and read from the same control register during the same 125 nanosecond cycle interval. Such an attempt is termed a conflict. If a potential conflict is not detected and resolved in the control section, the writing operation is performed properly but the results of the read operation are not defined. An undetected conflict may involve the control register references required for two consecutive instructions or the references required within certain individual instructions.

An analysis of the steps performed during the execution of two sequential instructions frequently indicates an anomaly; the result produced by the second instruction in the sequence depends on whether the first instruction is performed with alternate bank or with same bank timing. Even if the coding for two sequential instructions seems to indicate alternate bank timing for the first instruction, an anomaly may exist in that the results produced by the second instruction may depend on whether or not an I/O transfer sequence or an interrupt occurs between the two instructions.

Special circuitry in the control section detects and resolves, or eliminates most potential anomalies and conflicts. There are some instances, however, in which an anomaly or conflict is not detected by the control section.

5.4.3.1. Detected A or A+1/X Anomalies

When an instruction is performed with alternate bank timing, the Read Xx step for the next instruction is usually performed before the results for the current instruction are stored in the register or registers addressed by the a field of the current instruction. Sometimes the control register addressed by the contents of either the F3 register or F4 register (A or A+1) for the current instruction is the same register as that addressed by the x field of the next instruction. For most instructions, when this is detected, the control section aborts the operation of the T0 chain for the next instruction and restarts it at cycle 1 immediately following completion of cycle 3 of the T1 chain. Consequently, the result words from the current instruction are stored before the Read Xx step in cycle 1 of the T0 chain for the next instruction is effectively performed. This introduces a delay of 375 nanoseconds, but it ensures that the result of the current instruction is in the X register before the register is used for the next instruction, regardless of whether alternate bank or same bank timing applies for the current instruction.

5.4.3.2. Undetected A,A+1, and A+2/X Anomalies

A control register used to store a result word for any of the following eight instructions must not be used as the index register addressed by the x field of the next instruction. For these eight instructions, an anomaly will not be detected by the control circuitry.

- Double Precision Floating Add
- Double Precision Floating Add Negative
- Double Precision Floating Multiply
- Double Precision Floating Divide

- Double Load And Unpack Floating
- Double Load And Convert To Floating
- Floating Expand And Load
- Floating Compress And Load

When any of the above eight instructions are executed using alternate bank timing, the Write Aa, Write Aa+1, and Write Aa+2 steps are performed after the Read Xx step for the next instruction. When executing any of them with same bank timing, the necessary write steps are performed before the Read Xx step for the next instruction. Furthermore, if the instruction appears to provide alternate bank timing but an interrupt sequence is initiated immediately following the completion of the second pass through the T0 chain for that instruction, the write steps for the instruction are performed before rather than after the Read Xx step for the next sequential instruction.

5.4.3.3. Detected A/A Conflicts

When an instruction is executed using alternate bank timing, the writing of a result word in a control register for the current instruction during cycle 3 of the T1 chain normally occurs during the same 125 nanosecond interval as the reading of a control register to provide an input word to the arithmetic section (cycle 3 of the T0 chain) for the next instruction. The control section detects situations which call for reading from and writing in the same control register during cycle 3 of the T0 and T1 chains. When it detects such a situation, it does not perform the read operation. Instead, it initiates the needed write operation for the current instruction using the data word supplied by the arithmetic section. It also uses the data word in the arithmetic section as the input word to the next instruction. This is done without affecting the execution time of either instruction.

5.4.3.4. Undetected A+2/X Conflicts

The control register (Aa+2) used to store the count of the number of bit positions shifted during the execution of the Double Load Shift And Count (DLSC) instruction must not be used as the index register addressed by the x field of the next instruction. The DLSC instruction stores a result word in a control register during each of the three cycles of the T1 sequence. The count of the number of bit positions shifted is stored in control register Aa+2 during cycle 1. The index register for the next instruction is read during cycle 1 of the T0 chain (Read Xx). There is no conflict if control register Aa+2 for the DLSC instruction and the control register addressed by the x field of the next instruction are not the same and if same bank timing applies for the DLSC instruction, or if an interrupt occurs immediately following the DLSC instruction. If alternate bank timing applies for the DLSC instruction and control register Aa+2 for the DLSC instruction is the one which is addressed by the x field of the next instruction, the conflict is not detected. This occurs only if the a field of the DLSC instruction contains 00_8 and the x field of the next instruction contains 16_8 or if the two fields contain 01_8 and 17_8, respectively. If this situation arises, the Write Aa+2 is always properly performed but the input to the index subsection as a result of the Read Xx step for the next instruction is undefined.

5.4.3.5. Undetected X/A Conflicts

If the h field contains a 1 bit, the x field and the a field must not specify the same control register for the following three instructions:

- Logical AND
- Test Even Parity
- Test Odd Parity

For most instructions the Read Aa step, if required, is performed in cycle 3 of the T0 chain, and the Write Xx step (store the sum of Xm + Xi if h = 1, x ≠ 0) is performed in cycle 4 of the T0 chain. Both the Read Aa step and Write Xx step (if required), however, are performed in cycle 4 of the T0 chain for the above three instructions. There is no conflict if the x field and the a field of one of these instructions specify different control registers, or if h = 0. If h=1, and the x and a fields of any one of these instructions specify the same control register, an undetected conflict occurs during cycle 4 of the T0 chain.

5.4.3.6. Undetected X/A+1 Conflicts

If h = 1, the x field and the contents of the a field plus 1 must not specify the same control register for the following ten instructions:

- Divide Integer
- Divide Fractional
- Test Within Range
- Test Not Within Range
- Double Precision Zero Jump
- Double Shift Circular
- Double Shift Logical
- Double Shift Algebraic
- Left Double Shift Circular
- Left Double Shift Logical

Many instructions which require both a Read Aa step and a Read Aa+1 step require two passes through the T0 chain. For these instructions, the Read Aa step is performed during cycle 3 of the first pass through the T0 chain and the Read Aa+1 step is performed during cycle 3 of the second pass. The above ten instructions require only one pass through the T0 chain even though they require both a Read Aa step and a Read Aa+1 step. The Read Aa step is performed in cycle 3 and the Read Aa+1 step is performed in cycle 4. If h = 1 and the x field of the instruction specifies the same control register as the contents of the a field plus 1, an undetected conflict occurs in cycle 4 of the T0 chain.

There is no conflict if h = 0 or if the x field of the instruction does not specify the same control register as the contents of the a field plus 1.

6. INSTRUCTION REPERTOIRE

6.1. INTRODUCTION

This section describes the operation performed by each instruction in the UNIVAC 1108 repertoire. These descriptions are grouped by types of instructions. The descriptions assume a thorough understanding of the information presented in Section 5.

An introductory paragraph to each group presents information that is common to all instructions in the group. The detailed descriptions that follow an introductory paragraph have the following format:

- Instruction name
- Mnemonic code
- Octal function code
- Instruction execution times for alternate and same main storage module operation.
- Symbolic description of the operation performed by the instruction. The symbology used is defined in Appendix A.
- Textual description of the operation performed by the instruction.
- Sequentially numbered notes which provide special information related to the instruction, as appropriate.

For all instructions, any possible value may be used in the a, x, h, i, and u fields unless an exception to this rule is stated in the notes. Any possible value may be used in the j field except when j is a minor function code designator or when an exception is stated in the notes.

6.2. LOAD INSTRUCTIONS

The single-precision load instructions transfer data to the arithmetic section where a 36-bit word is always formed. The 36-bit word is then transferred to the control register specified by the a field of the instruction. Single-precision data-word transfers from main storage to the arithmetic section are controlled by the value in the j field.

For the double-precision load instructions, the j field is a minor function code and full 72-bit data transfers result.

A word consisting of all 0 bits represents +0 and a word consisting of all 1 bits represents −0.

6.2.1. Load A

L,LA
10
0.75 μs alternate
1.50 μs same

(U)→A

The contents of U are transferred under j field control to the arithmetic section and then to Aa.

(1) A Load A instruction cannot load a half word or a full word consisting or all 1 bits using j = 16_8 or 17_8, and h, i, u = −0. See the Load Negative A instruction (6.2.2) or the Load Negative Magnitude A instruction (6.2.4) for loading a full word of all 1 bits. For the Load A instruction, if j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0 and the U value transferred to the arithmetic section is +0.

6.2.2. Load Negative A

LN,LNA
11
0.75 μs alternate
1.50 μs same

−(U) → A

The contents of U are transferred under j field control to the arithmetic section. The ones complement of the value in the arithmetic section is transferred to Aa.

(1) This instruction may be used to load −0 into an A register by using j = 16_8 or 17_8, and x = h = i = u = 0.

6.2.3. Load Magnitude A

LM,LMA
12
0.75 μs alternate
1.50 μs same

|(U)| → A

The contents of U are transferred under j field control to the arithmetic section. If the sign bit (bit 35) of the value in the arithmetic section is a 1 bit, it is complemented; if the sign bit is a 0 bit, it is not complemented. The final value (always positive) is transferred from the arithmetic section to Aa.

6.2.4. Load Negative Magnitude A

LNMA
13
0.75 μs alternate
1.50 μs same

− |(U)| → A

The contents of U are transferred under j field control to the arithmetic section. If the sign bit (bit 35) of the value in the arithmetic section is a 0 bit, it is complemented; if the sign bit is a 1 bit, it is not complemented. The final value (always negative) is transferred from the arithmetic section to Aa.

(1) This instruction may be used to load −0 into an A register by using $j = 16_8$ or 17_8, and x = h = i = u = 0.

6.2.5. Load R

L,LR
23
0.75 μs alternate
1.50 μs same

$(U) \rightarrow R_a$

The contents of U are transferred under j field control to the arithmetic section and then to the R register specified by the a field.

(1) It is not possible to load a half word or a full word of 1 bits into an R register using this instruction with $j = 16_8$ or 17_8, x = 0, and h, i, u = −0.

(2) If the CPU is in guard mode, an attempt to Load R0 causes a Guard Mode Fault Interrupt.

6.2.6. Load X Modifier

LXM
26
0.875 μs alternate
1.625 μs same

$(U) \rightarrow X_{a_{17-0}}$; $X_{a_{35-18}}$ unchanged

The contents of U are transferred under j field control to the arithmetic section; the low order 18 bits of the value in the arithmetic section are transferred to the lower half (bits 17–0) of the X register specified by the a field; the upper half (bits 35 through 18) of the X register remains unchanged.

(1) It is not possible to load the lower half of an X register with all 1 bits using this instruction with $j = 16_8$ or 17_8, x = 0, and h, i, u = −0.

6.2.7. Load X

L,LX
27
0.75 μs alternate
1.50 μs same

$(U) \rightarrow X_a$

The contents of U are transferred under j field control to the arithmetic section and then to the X register specified by the a field.

(1) It is not possible to load an X register with a half word or full word of all 1 bits using this instruction with j = 16_8 or 17_8, x = 0 and h, i, u = −0.

6.2.8. Load X Increment

LXI
46
1.00 μs alternate
1.75 μs same

(U) → $X_{a_{35-18}}$; $X_{a_{17-0}}$ unchanged

The contents of U are transferred under j field control to the arithmetic section; the low order 18 bits of the value in the arithmetic section are transferred to the upper half (bits 35–18) of the X register specified by the a field. The lower half (bits 17–0) of the X register remains unchanged.

(1) It is not possible to load the upper half of an X register with all 1 bits using this instruction with j = 16_8 or 17_8, x = 0, and h, i, u = −0.

6.2.9. Double Load A

DL
71,13
1.50 μs alternate
2.25 μs same

(U,U+1) → A,A+1

The contents of U and U+1 are transferred to the arithmetic section and then to Aa and Aa+1, respectively.

(1) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.2.10. Double Load Negative A

DLN
71,14
1.50 μs alternate
2.25 μs same

−(U,U+1) → A,A+1

The contents of U and U+1 are transferred to the arithmetic section where the 72-bit value is complemented and then transferred to Aa and Aa+1, respectively.

(1) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.2.11. Double Load Magnitude A

DLM
71,15
1.50 μs alternate
2.25 μs same

$|(U,U+1)| \rightarrow A,A+1$

The contents of U and U+1 are transferred to the arithmetic section. If the sign bit (bit 35) of U is a 1 bit, the 72-bit value in the arithmetic section is complemented; if the sign bit is a 0 bit, the 72-bit value is not complemented. The final value (always positive) is transferred from the arithmetic section to Aa and Aa+1.

(1) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.3. STORE INSTRUCTIONS

The single-precision store instructions transfer data from a control register specified by the a field to the arithmetic section, where the data is operated on, if required, and then transferred to the main storage location or control register addressed by U. An exception to this is the Store Zero instruction (see 6.3.5) which causes zeros to be transferred from the arithmetic section to location U.

Single-precision data-word transfers to main storage are controlled by the j field as explained in 5.2.2.1 and illustrated in Figure 5–3. The j field cannot be used to transfer data from one control register to another control register ($U < 200_8$) and only full word (36-bit) data can be transferred. If j = 16_8 or 17_8, no data is stored. A Guard Mode Fault Interrupt will occur, however, if $U < 200_8$ and the guard mode is violated, or if $U \geq 200_8$ and SI or SD, as appropriate, is outside the main storage limits set in the Storage Limits Register. The j field of the double-precision store instruction is a minor function code and a full 72-bit data transfer results.

Indexing, index incrementation/decrementation, and indirect addressing function normally in *all* cases.

A word consisting of all 0 bits represents +0 and a word consisting of all 1 bits represents –0.

6.3.1. Store A

S,SA
01
0.75 μs alternate
1.50 μs same

$(A) \rightarrow U$

The contents of Aa are transferred to the arithmetic section and then transferred under j field control to location U.

(1) If j = 16_8 or 17_8, no data is stored.

6.3.2. Store Negative A

SN,SNA
02
0.75 μs alternate
1.50 μs same

$-(A) \rightarrow U$

The contents of Aa are transferred to the arithmetic section. The complement of the value in the arithmetic section is transferred under j field control to location U.

(1) If j = 16_8 or 17_8, no data is stored.

6.3.3. Store Magnitude A

SM,SMA
03
0.75 μs alternate
1.50 μs same

$|(A)| \rightarrow U$

The contents of Aa are transferred to the arithmetic section. If the sign bit (bit 35) of the value in the arithmetic section is a 1 bit, the value is complemented. The final value (always positive) is transferred from the arithmetic section under j field control to location U.

(1) If j = 16_8 or 17_8, no data is stored.

6.3.4. Store R

S,SR
04
0.75 μs alternate
1.50 μs same

$(R_a) \rightarrow U$

The contents of the R register specified by the a field are transferred to the arithmetic section and then transferred under j field control to location U.

(1) If j = 16_8 or 17_8, no data is stored.

6.3.5. Store Zero

SZ
05
0.75 μs alternate
1.50 μs same

Zeros $\rightarrow$ U

Binary zeros are transferred under j field control from the arithmetic section to location U.

(1) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(2) If j = 16_8 or 17_8, no data is stored.

6.3.6. Store X

S,SX
06
0.75 μs alternate
1.50 μs same

$(X_a) \rightarrow U$

The contents of the X register specified by the a field are transferred to the arithmetic section and then transferred under j field control to location U.

(1) If j = 16_8 or 17_8, no data is stored.

6.3.7. Double Store A

DS
71,12
1.50 μs alternate
2.25 μs same

$(A,A+1) \rightarrow U,U+1$

The contents of Aa and Aa+1 are transferred to the arithmetic section and then to locations U and U+1, respectively.

(1) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.3.8. Block Transfer

BT
22
2.25 + 1.5K μs
K = the initial count in the Repeat Count Register

$(X_x + u) \rightarrow X_a + u$; repeat K times.

A source word is transferred under j field control to a nonaddressable register in the arithmetic section. The contents of the nonaddressable register are transferred under j field control to a destination word location. The repeat count is decremented by 1. The source to destination transfer step is repetitively performed until the repeat count has been decremented to 0. The x field specifies the X register used with the u field to determine the effective source word address. The a field specifies the X register used in determining the effective destination word address.

(1) A word containing the desired repeat count in the rightmost 18-bit positions must be loaded in the Repeat Count Register (R1) before performing the Block Transfer instruction. It is recommended that this word contain all 0 bits in the leftmost 18-bit positions.

(2) Instruction initiation requires a .75 microsecond pass through the main timing chain (T0) to transfer the contents of R1 to the index subsection. Two passes through T0 are required for each data word transfer. A .75 microsecond termination pass through T0 stores the remnant repeat count in R1 after all specified transfer steps have been completed (repeat count has decremented to zero) or in preparation for an I/O interrupt.

(3) If the initial repeat count is +0, only the initiation and termination passes occur and no data is transferred.

(4) If j = 16_8 or 17_8, no data is transferred; however, the data transfer passes will occur until the repeat count is decremented to zero.

(5) If the x field is zero, no data is transferred. The contents of the X register specified by the a field remain unchanged regardless of the contents of the a and h fields.

(6) The P register is incremented by 1 during the initial data transfer pass. If I/O interrupt occurs before the repeat count has decremented to zero, the termination pass occurs at the conclusion of the currently active data transfer pass. The remnant repeat count is stored in R1 and the contents of the P register are decremented by 1 during this termination pass. When the instruction at the interrupt location is initiated, the P register contains the address of the Block Transfer instruction or the address of the Execute instruction which led to the Block Transfer instruction. Thus this address can be preserved and, when the interrupt condition has been satisfied, it is possible to return to the Block Transfer instruction and continue executing this instruction at the point where it was terminated for the interrupt. If the Block Transfer instruction was entered by means of an Execute instruction, the h field of the Execute instruction should be zero so that, when the program returns to the Execute instruction, the effective U address will again lead to the Block Transfer instruction. If the Block Transfer instruction contains indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and complete the Block Transfer instruction in the event of an interrupt.

(7) If there is no indirect addressing (i field = 0), the h field normally contains a 1 bit. If h = 0, no incrementation/decrementation of the index registers occurs. The source and destination addresses when h = 0 are the initial contents of the index registers used repetitively for every transfer perform. Thus no more than one data transfer is effectively performed.

(8) In user operating mode (D6 of the PSR = 0), if the x field is not zero, but the a field is zero, the a field references the PSR Temporary Storage register at control register address 000_8. If an interrupt should occur during the execution of such a Block Transfer instruction, the contents of the PSR are stored in the PSR Temporary Storage register. Thus the a field reference for the Block Transfer instruction is destroyed and there is no way to return to complete the Block Transfer instruction when the interrupt condition has been satisfied.

In the Executive operating mode (D6 of the PSR = 1), if the x field is not zero and the a field is zero, the a field references the Executive Nonindexing Index register at control register address 140_8 (see 3.2.2.14) and proper operation results.

In either user or Executive mode, the contents of the control register referenced when a = 0 are used by the index subsection to determine the destination address during the second pass through the T0 chain for each data transfer pass, and incrementation/decrementation of index registers operates normally.

(9) If j = 1_8 to 15_8 and a source address developed in the index subsection is a control register address, the source to arithmetic portion of the pass is performed as if j = 0. A full 36-bit word is effectively transferred from the control register to the arithmetic section. The transfer from the arithmetic section to the destination address is also under j field control.

Thus the j field can specify partial word transfers from the arithmetic section to main storage addresses. If the destination address is a control register address, the full 36-bit word is transferred from the arithmetic section to the specified control register.

6.4. Fixed-Point Arithmetic Instructions

The fixed-point arithmetic instructions perform integer or fractional addition, subtraction, multiplication, and division. Details of the operation of the arithmetic section are explained in 4.1 through 4.3.6. In a single-precision arithmetic instruction, the transfer of data from location U in main storage to the arithmetic section is under the control of the contents of the j field of the instruction as illustrated in Figure 5–2. For double-precision and parallel half word and third word arithmetic operations, the value in the j field is a minor function code.

For all arithmetic instructions, indexing, index incrementation/decrementation, and indirect addressing function normally.

A word consisting of all 0 bits represents +0 and a word consisting of all 1 bits represents −0.

6.4.1. Add To A

A,AA
14
0.75 μs alternate
1.50 μs same

(A) + (U) → A

The contents of U are transferred under j field control to the arithmetic section. The 36-bit value in the arithmetic section is added algebraically to the contents of Aa. The sum is stored in Aa.

(1) The overflow and carry designators are cleared at initiation and may be set as a result of performing this instruction.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

(3) $-0 + -0 = -0$
$-0 + +0 = +0$
$+0 + -0 = +0$
$+0 + +0 = +0$

6.4.2. Add Negative To A

AN,ANA
15
0.75 μs alternate
1.50 μs same

(A) – (U) → A

The contents of U are transferred under j field control to the arithmetic section. The 36-bit value in the arithmetic section is subtracted algebraically from the contents of Aa. The difference is stored in Aa.

(1) The overflow and carry designators are cleared at initiation and may be set as a result of performing this instruction.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than –0 and the U value transferred to the arithmetic section is +0.

(3) $-0 - +0 = -0$
$-0 - -0 = +0$
$+0 - -0 = +0$
$+0 - +0 = +0$

6.4.3. Add Magnitude To A

AM,AMA
16
0.75 μs alternate
1.50 μs same

(A) + |(U)| → A

The contents of U are transferred under j field control to the arithmetic section. If the sign bit (bit 35) of the 36-bit value in the arithmetic section is a 1 bit, the value is complemented; if the sign bit is a 0 bit, the value is not complemented. The final 36-bit value in the arithmetic section (always positive) is added algebraically to the contents of Aa. The sum is stored in Aa.

(1) The overflow and carry designators are cleared at initiation and may be set as a result of performing this instruction.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than –0, and the U value transferred to the arithmetic section is +0.

(3) $-0 + +0 = +0$
$+0 + +0 = +0$

6.4.4. Add Negative Magnitude To A

ANM,ANMA
17
0.75 μs alternate
1.50 μs same

(A) -- |(U)| → A

The contents of U are transferred under j field control to the arithmetic section. If the sign bit (bit 35) of the 36-bit value in the arithmetic section is a 1 bit, the value is complemented; if the sign bit is a 0 bit, the value is not complemented. The final 36-bit value in the arithmetic section (always positive) is subtracted algebraically from the contents of Aa. The difference is stored in Aa.

(1) The overflow and carry designators are cleared at initiation and may be set as a result of performing this instruction.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

(3) −0 − +0 = −0
+0 − +0 = +0

6.4.5. Add Upper

AU
20
0.75 μs alternate
1.50 μs same

(A) + (U) → A+1

The contents of U are transferred under j field control to the arithmetic section. The 36-bit value in the arithmetic section is added algebraically to the contents of Aa. The sum is stored in Aa+1. The contents of U and Aa remain unchanged.

(1) The overflow and carry designators are cleared at initiation and may be set as a result of performing this instruction.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

(3) −0 + −0 = −0
−0 + +0 = +0
+0 + −0 = +0
+0 + +0 = +0

(4) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8

6.4.6. Add Negative Upper

ANU
21
0.75 μs alternate
1.50 μs same

(A) – (U) → A+1

The contents of U are transferred under j field control to the arithmetic section. The 36-bit value in the arithmetic section is subtracted algebraically from the contents of Aa. The difference is stored in Aa+1. The contents of U and Aa remain unchanged.

(1) The overflow and carry designators are cleared at initiation and may be set as a result of performing this instruction.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than –0, and the U value transferred to the arithmetic section is +0.

(3) –0 – +0 = –0
–0 – –0 = +0
+0 – –0 = +0
+0 – +0 = +0

(4) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.4.7. Add To X

A,AX
24
0.75 μs alternate
1.50 μs same

$(X_a) + (U) \rightarrow X_a$

The contents of U are transferred under j field control to the arithmetic section. The 36-bit value in the arithmetic section is added algebraically to the contents of the X register specified by the a field. The sum is stored in the X register specified by the a field.

(1) The overflow and carry designators are cleared at initiation and may be set as a result of performing this instruction.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than –0, and the U value transferred to the arithmetic section is +0.

(3) –0 + –0 = –0
–0 + +0 = +0
+0 + –0 = +0
+0 + +0 = +0

6.4.8. Add Negative To X

AN,ANX
25
0.75 μs alternate
1.50 μs same

$(X_a) - (U) \rightarrow X_a$

The contents of U are transferred under j field control to the arithmetic section. The 36-bit value in the arithmetic section is subtracted algebraically from the contents of the X register specified by the a field. The difference is stored in the X register specified by the a field.

(1) The overflow and carry designators are cleared at initiation and may be set as a result of performing this instruction.

(2) If $j = 16_8$ or 17_8, $x = 0$, $h = i = 1$, and $u = 177777_8$, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

(3) −0 − +0 = −0
−0 − −0 = +0
+0 − −0 = +0
+0 − +0 = +0

6.4.9. Multiply Integer

MI
30
2.375 μs alternate
3.125 μs same

$(A) \cdot (U) \rightarrow A,A+1$

The contents of U are transferred under j field control to the arithmetic section. The contents of Aa are multiplied algebraically by the 36-bit value in the arithmetic section, producing a 72-bit product. The most significant 36 bits of the product (including sign bits) are stored in Aa. The least significant 36 bits of the product are stored in Aa+1.

(1) Bit positions 71 and 70 of the product are always sign bits. The product of any two 35-bit positive integers cannot exceed a 70-bit positive integer.

(2) The algebraic rule for signs applies without exception.

(3) If $j = 16_8$ or 17_8, $x = 0$, $h = i = 1$, and $u = 177777_8$, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

(4) If $a = 17_8$, Aa+1 is the control register at address 34_8 or 174_8.

6.4.10. Multiply Single Integer

MSI
31
2.375 μs alternate
3.125 μs same

(A) · (U) → A

The contents of the U are transferred under j field control to the arithmetic section. The contents of Aa are multiplied algebraically by the 36-bit value in the arithmetic section, producing a 72-bit product. The least significant 36 bits of the product are stored in Aa. The most significant 36 bits of the product are lost.

(1) The 36-bit result stored in Aa does not represent the product as a signed number if the leftmost 37 bits of the 72-bit product formed in the arithmetic section are not identical.

(2) The algebraic rule for signs applies except as noted above.

(3) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

6.4.11. Multiply Fractional

MF
32
2.375 μs alternate
3.125 μs same

(A) · (U) → A,A+1

The contents of U are transferred under j field control to the arithmetic section. The contents of Aa are multiplied algebraically by the 36-bit value in the arithmetic section, producing a 72-bit product which is shifted left, circularly, one bit position. The leftmost 36 bits of the shifted product, including the sign bit are stored in Aa. The rightmost 36 bits are stored in Aa+1.

(1) This instruction performs an operation identical to the Multiply Integer instruction (see 6.4.9) except that the 72-bit result of the multiplication process is shifted left, circularly, one bit position prior to storing it in Aa and Aa+1.

(2) The algebraic rule for signs applies without exception.

(3) The rightmost bit of the result in Aa+1 is a sign bit and it is identical to the leftmost bit of the result in Aa.

(4) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

(5) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.4.12. Divide Integer

DI
34
10.125 μs alternate
10.875 μs same

(A,A+1) ÷ (U) → A; remainder → A+1

The contents of U are transferred under j field control to the arithmetic section. The 72-bit signed number in Aa and Aa+1 is shifted one bit position left, circularly, and divided algebraically by the 36-bit value in the arithmetic section. The 36-bit signed quotient is stored in Aa. The remainder retains the sign of the dividend (the leftmost bit of the initial contents of Aa) and is stored in Aa+1.

(1) The absolute value of the 72-bit signed dividend (Aa,Aa+1) should be less than the absolute value of the divisor (j-determined portion of U) multiplied by 2^{35}. If this relationship is not satisfied, no result is stored (the initial contents of Aa and Aa+1 are not changed) and a Divide Fault Interrupt results. This includes the case in which the divisor equals ±0.

(2) The algebraic rule for signs applies to the sign of the quotient (see 4.3.5).

(3) When h = 1, the value transferred to the arithmetic section from Aa+1 is undefined if Aa+1 is also the X register being incremented; that is, for the following three combinations of x and a field values:

$x = 15_8$ and a = 0
$x = 16_8$ and a = 1
$x = 17_8$ and a = 2

(4) If $j = 16_8$ or 17_8, x = 0, h = i = 1, and $u = 177777_8$, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

(5) If $a = 17_8$, Aa+1 is the control register at address 34_8 or 174_8.

6.4.13. Divide Single Fractional

DSF
35
10.125 μs alternate
10.875 μs same

(A) ÷ (U) → A+1

The contents of U are transferred under j field control to the arithmetic section. The contents of Aa are divided algebraically by the 36-bit value in the arithmetic section. The 36-bit signed quotient is stored in Aa+1. The remainder is lost. The contents of Aa remain unchanged.

(1) The absolute value of the dividend (Aa) should be less than the absolute value of the divisor (j-determined portion of U). If this relationship is not satisfied, no result is stored (the initial contents of Aa+1 are not changed), and a Divide Fault Interrupt results. This includes the case in which the divisor equals ±0.

(2) The algebraic rule for signs applies to the sign of the quotient (see 4.3.5).

(3) If $j = 16_8$ or 17_8, $x = 0$, $h = i = 1$, and $u = 177777_8$, the 18-bit output of the index adder is +0 rather than −0. As a consequence, the U value transferred to the arithmetic section is +0 and a Divide Fault Interrupt occurs.

(4) If $a = 17_8$, Aa+1 is the control register at address 34_8 or 174_8.

6.4.14. Divide Fractional

DF
36
10.125 μs alternate
10.875 μs same

(A,A+1) ÷ (U) → A; remainder → A+1

The contents of U are transferred under j field control to the arithmetic section. The 72-bit signed number in Aa and Aa+1 is divided algebraically by the 36-bit value in the arithmetic section. The 36-bit signed quotient is stored in Aa. The remainder retains the sign of the dividend (the leftmost bit of the original contents of Aa) and is stored in Aa+1.

(1) If the absolute value of the leftmost half of the dividend (Aa) is greater than, or equal to, the absolute value of the divisor, no result is stored (the initial contents of Aa and Aa+1 remain unchanged), and a Divide Fault Interrupt occurs. This includes the case in which the divisor equals ±0.

(2) When $h = 1$, the value transferred to the arithmetic section from Aa+1 is undefined if Aa+1 is also the X register being incremented; that is, for the following three combinations of x and a field values:

$x = 15_8$ and $a = 0$
$x = 16_8$ and $a = 1$
$x = 17_8$ and $a = 2$

(3) The algebraic rule for signs applies to the sign of the quotient (see 4.3.5).

(4) If $j = 16_8$ or 17_8, $x = 0$, $h = i = 1$, and $u = 177777_8$, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

(5) If $a = 17_8$, Aa+1 is the control register at address 34_8 or 174_8.

6.4.15. Double-Precision Fixed Point Add

DA
71,10
1.625 μs alternate
2.375 μs same

(A,A+1) + (U,U+1) → A,A+1

The 72-bit signed number from U and U+1 is added algebraically to the 72-bit signed number from Aa and Aa+1. The 72-bit sum is stored in Aa and Aa+1.

(1) The overflow and carry designators are cleared at initiation and may be set as a result of performing this instruction.

(2) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

(3) −0 + −0 = −0
−0 + +0 = +0
+0 + +0 = +0
+0 + −0 = +0

6.4.16. Double-Precision Fixed Point Add Negative

DAN
71,11
1.625 μs alternate
2.375 μs same

(A,A+1) − (U,U+1) → A,A+1

The 72-bit signed number from U and U+1 is subtracted algebraically from the 72-bit signed number from Aa and Aa+1. The 72-bit difference is stored in Aa and Aa+1.

(1) The overflow and carry designators are cleared at initiation and may be set as a result of performing this instruction.

(2) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

(3) −0 − +0 = −0
−0 − −0 = +0
+0 − +0 = +0
+0 − −0 = +0

6.4.17. Add Halves

AH
72,04
0.75 μs alternate
1.50 μs same

$(A)_{35-18} + (U)_{35-18} \rightarrow A_{35-18}$;
$(A)_{17-0} + (U)_{17-9} \rightarrow A_{17-0}$

The contents of each half (18-bit portion) of U are added algebraically to the contents of the corresponding half of Aa. The sums are stored in the corresponding halves of Aa.

(1) The overflow and carry designators are not affected by this instruction.

(2) There is no interaction between the upper and lower halves in the arithmetic section. A borrow from bit 17 is propagated to bit 0 rather than bit 18. A borrow from bit 35 is progated to bit 18 rather than bit 0.

(3) $-0 + -0 = -0$
$-0 + +0 = +0$
$+0 + -0 = +0$
$+0 + +0 = +0$

6.4.18. Add Negative Halves

ANH
72,05
0.75 μs alternate
1.50 μs same

$(A)_{35-18} - (U)_{35-18} \rightarrow A_{35-18}$;

$(A)_{17-0} - (U)_{17-0} \rightarrow A_{17-0}$

The contents of each half (18-bit portion) of U are subtracted algebraically from the contents of the corresponding half of Aa. The differences are stored in the corresponding halves of Aa.

(1) The overflow and carry designators are not affected by this instruction.

(2) There is no interaction between the upper and lower halves in the arithmetic section. A borrow from bit 17 is propagated to bit 0 rather than bit 18. A borrow from bit 35 is propagated to bit 18 rather than bit 0.

(3) $-0 - +0 = -0$
$-0 - -0 = +0$
$+0 - +0 = +0$
$+0 - -0 = +0$

6.4.19. Add Thirds

AT
72,06
0.75 μs alternate
1.50 μs same

$(A)_{35-24} + (U)_{35-24} \rightarrow A_{35-24}$;

$(A)_{23-12} + (U)_{23-12} \rightarrow A_{23-12}$;

$(A)_{11-0} + (U)_{11-0} \rightarrow A_{11-0}$

The contents of each third (12-bit portion) of U are added algebraically to the contents of the corresponding third of Aa. The sums are stored in the corresponding thirds of Aa.

(1) The overflow and carry designators are not affected by this instruction.

(2) A borrow from bit 11, 23, or 35 is propagated to bit 0, 12, or 24, respectively, rather than to bit 12, 24, or 0.

(3) $-0 + -0 = -0$
$-0 + +0 = +0$
$+0 + +0 = +0$
$+0 + -0 = +0$

6.5. FLOATING-POINT ARITHMETIC

Floating-point arithmetic operations allow for efficient computation involving numerical data with a wide range of magnitudes (see 4.4). In all floating-point arithmetic instructions, indexing, index incrementation/decrementation, and indirect addressing function normally.

The greatest precision is obtained in floating-point arithmetic operations when the floating-point input operands are normalized numbers. Certain floating-point operations produce undefined results if normalized input operands are not used; the supporting notes indicate which instructions are affected. A word consisting of all 0 bits represents +0 and a word consisting of all 1 bits represents −0.

6.5.1. Floating Add

FA
76,00
1.875 μs alternate
2.626 μs same

(A) + (U) → A; residue → A+1

The single-precision floating-point number from location U is added to the single-precision floating-point number from Aa. The resulting sum is normalized and then stored in single-precision floating-point format in Aa. The residue in single-precision floating-point format is stored in Aa+1.

(1) The result stored in Aa is a normalized number even if either or both of the input operands are not normalized. No attempt is made to normalize the residue stored in Aa+1.

(2) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt may occur (see 4.4.3).

(3) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.5.2. Floating Add Negative

FAN
76,01
1.875 μs alternate
2.625 μs same

(A) – (U) → A; residue → A+1

The single-precision floating-point number from location U is subtracted from the single-precision floating-point number from Aa. The resulting difference is normalized and then stored in single-precision floating-point format in Aa. The residue in single-precision floating-point fromat is stored in Aa+1.

(1) The result stored in Aa is a normalized number even if either or both of the input operands are not normalized. No attempt is made to normalize the residue stored in Aa+1.

(2) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt may occur (see 4.4.3).

(3) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

(4) The Floating Add Negative operation performed in the arithmetic section is identical to the Floating Add operation described in 6.5.1 except that the ones complement of the contents of location U is used as the second input operand.

6.5.3. Double-Precision Floating Add

DFA
76,10
2.625 μs alternate
3.375 μs same

(A,A+1) + (U,U+1) → A,A+1

The double-precision floating-point number from locations U and U+1 is added to the double-precision floating-point number from Aa and Aa+1. The resulting sum is normalized and then stored in double-precision floating-point format in Aa and Aa+1.

(1) The result stored is a normalized number even if either or both of the input operands are not normalized.

(2) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt may occur (see 4.4.3).

(3) If characteristic arithmetic produces a characteristic for the sum which represents a negative number and D5 of PSR = 1, a Floating-Point Characteristic Underflow Fault Interrupt does not occur. Instead, +0 is stored in Aa and Aa+1.

(4) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

(5) Neither of the A registers used to store the sum should be used as the index register specified by the x field of the next sequential instruction.

6.5.4. Double-Precision Floating Add Negative

DFAN
76,11
2.625 μs alternate
3.375 μs same

(A,A+1) – (U,U+1) → A,A+1

The double-precision floating-point number from locations U and U+1 is subtracted from the double-precision floating-point number from Aa and Aa+1. The resulting difference is normalized and then stored in double-precision floating-point format in Aa and Aa+1.

(1) The result stored is a normalized number even if either or both of the input operands are not normalized.

(2) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt may occur (see 4.4.3).

(3) If characteristic arithmetic produces a characteristic for the difference which represents a negative number and D5 of PSR = 1, a Floating-Point Characteristic Underflow Fault Interrupt does not occur. Instead, +0 is stored in Aa and Aa+1.

(4) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

(5) The Double Precision Floating Add Negative operation performed in the arithmetic section is identical to the Double Precision Floating Add process described in 6.5.3 except that the ones complement of the contents of U and U+1 is used as the second input operand.

(6) Neither of the A registers used to store the difference should be used as the index register specified by the x field of the next sequential instruction.

6.5.5. Floating Multiply

FM
76,02
2.625 μs alternate
3.375 μs same

(A) · (U) → A,A+1

The single-precision floating-point number from Aa is multiplied by the single-precision floating-point number from location U. The resulting double-length product is packed into two single-precision floating-point numbers. The most significant portion of the product in single-precision floating-point format is stored in Aa. The least significant portion of the product in single-precision floating-point format is stored in Aa+1.

(1) If either or both input operands are not normalized numbers, the results are undefined. The following notes apply only if both input operands are normalized numbers.

(2) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt may occur (see 4.4.3).

(3) The portion of the product stored in Aa is a normalized number. Except for the special case explained in note 6, no attempt is made to normalize the number stored in Aa+1.

(4) The algebraic rule for signs applies to the portions of the product stored in Aa and Aa+1 except for the special cases covered in notes 5b and 5d.

(5) If the mantissa of either or both input operands is zero, the following applies:

(a) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt never occurs, regardless of the values of the characteristics of the input operands.

(b) If D8 of PSR = 0, the result stored in Aa is +0 regardless of the signs of the input operands.

(c) If D8 = 1 and if the characteristic is in the range 000_8 through 377_8, the most significant product word will reflect the magnitude of the characteristic produced and the sign produced by the mantissa arithmetic.

If the characteristic arithmetic produces a characteristic for the most significant product word which represents a number greater than 377_8 or negative number, the result stored in Aa is +0 or −0 whichever would reflect the signs of the input operands.

(d) The value of D8 has no effect on the least significant product word. When the mantissa for the least significant product word is zero, it is packed with the appropriate characteristic unless the characteristic is not within representable range. If there is underflow on the characteristic, the word is set to +0 or −0 whichever would agree with the sign of the product. If the characteristic arithmetic for the least significant product word produces a characteristic which represents a number greater than 377_8, the full 9-bit characteristic adder output (which includes a 1 bit in the leftmost bit position rather than a 0 bit) is packed with the mantissa bits (all 0 bits) for the word. The word is complemented if the input operands have dissimilar signs. Thus the result stored in Aa+1 violates the algebraic rule for signs.

(6) When the mantissa for the least significant product word is not zero, it is packed with the appropriate characteristic unless the characteristic arithmetic for the least significant product word produces a characteristic which represents a negative number. In this case, the result stored in Aa+1 is +0 or −0 whichever would reflect the signs of the input operands.

(7) If the characteristics of the number stored in Aa is greater than or equal to 33_8, then the characteristic of the number stored in Aa+1 is 33_8 less than the characteristic in Aa.

(8) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.5.6. Double-Precision Floating Multiply

DFM
76,12
4.25 μs alternate
5.00 μs same

(A,A+1) · (U,U+1) → A,A+1

The double-precision floating-point number from Aa and Aa+1 is multiplied by the double-precision floating-point number from locations U and U+1. The product is normalized and stored in double-precision floating-point format in Aa and Aa+1.

(1) If either or both input operands are not normalized numbers, the results are undefined. The following notes apply only if both operands are normalized numbers.

(2) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt may occur (see 4.4.3).

(3) The result stored in Aa and Aa+1 is always a normalized number.

(4) The algebraic rule for signs applies except for the special cases covered in notes 5b and 6.

(5) If the mantissa of either of both input operands is zero, the following applies:

(a) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt never occurs, regardless of the values of the characteristics of the input operands.

(b) The result stored in Aa and Aa+1 is +0 regardless of the signs of the input operands.

(6) If the characteristic arithmetic produces a characteristic for the product which represents a negative number and D5 of PSR = 1, a Floating-Point Characteristic Underflow Fault Interrupt does not occur. Instead +0, regardless of the signs of the input operands, is stored in Aa and Aa+1.

(7) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

(8) Neither of the A registers used to store the product should be used as the index register specified by the x field of the next sequential instruction.

6.5.7. Floating Divide

FD
76,03
8.25 μs alternate
9.00 μs same

(A) ÷ (U) → A; remainder → A+1

The single-precision floating-point number from Aa is divided by the single-precision floating-point number from location U. The quotient is stored in Aa in single-precision floating-point format. The remainder is stored in Aa+1 in single-precision floating-point format.

(1) If either or both input operands are not normalized numbers, the results are not defined. The following notes apply only if both operands are normalized numbers.

(2) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt may occur (see 4.4.3).

(3) If the mantissa of the divisor (U) is zero, a Divide Fault Interrupt occurs.

(4) The quotient stored in Aa is a normalized number. Except for the special case explained in note 7, no attempt is made to normalize the remainder stored in Aa+1.

(5) The algebraic rule for signs applies to the quotient stored in Aa, except for the special cases explained in notes 6b and 6d.

(6) If the mantissa of the dividend (Aa) is zero but not the divisor (U), the following applies:

(a) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt never occurs, regardless of the characteristics of the input operands.

(b) If D8 of PSR = 0, the quotient stored in Aa is +0 regardless of the signs of the input operands.

(c) If D8 of PSR = 1 and the characteristic arithmetic produces a characteristic for the quotient which represents a number greater than 377_8 or a negative number, the quotient stored in Aa is +0 or −0 whichever would reflect the signs of the input operands.

(d) If the characteristic arithmetic for the remainder produces a characteristic which represents a number greater than 377_8, the full 9-bit characteristic adder output (which includes a 1 bit in its leftmost bit position rather than a 0 bit) is packed with the mantissa bits (all 0 bits) for the remainder. The ones complement of this packed word is formed if the sign bit of the dividend from Aa was a 1 bit. The remainder stored in Aa+1 violates the algebraic rule for signs.

(7) If the characteristic arithmetic produces a characteristic for the remainder which represents a negative number, the remainder stored in Aa+1 is +0 or −0 whichever would reflect the sign of the dividend from Aa.

(8) If the characteristic of the dividend from Aa is greater than, or equal to, 33_8 then the characteristic of the number stored in Aa+1 for the remainder is 33_8 or 32_8 less than the characteristic of the dividend.

(9) If the absolute value of the dividend mantissa is less than the absolute value of the divisor mantissa, the instruction execution time is extended by .25 microseconds.

(10) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.5.8. Double-Precision Floating Divide

DFD
76,13
17.25 μs alternate
18.00 μs same

(A,A+1) ÷ (U,U+1) → A,A+1

The double-precision floating-point number from Aa and Aa+1 is divided by the double-precision floating-point number from locations U and U+1. The quotient is stored in Aa and Aa+1 in double-precision floating-point format. The remainder is not retained.

(1) If either or both of the input operands are not normalized numbers, the results are undefined. The following notes apply only if both operands are normalized numbers.

(2) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt may occur (see 4.4.3).

(3) If the mantissa of the divisor is zero, a Divide Fault Interrupt occurs.

(4) The result stored in Aa and Aa+1 is always a normalized number.

(5) The algebraic rule for signs applies except for the special cases explained in notes 6b and 7.

(6) If the dividend mantissa (Aa,Aa+1) is zero and the divisor mantissa (U,U+1) is not zero, the following applies:

(a) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt never occurs regardless of the values of the characteristics of the input operands.

(b) The result stored in Aa and Aa+1 is +0 regardless of the signs of the input operands.

(7) If the characteristic arithmetic produces a characteristic for the quotient which represents a negative number and D5 of PSR = 1, a Floating-Point Characteristic Underflow Fault Interrupt does not occur. Instead +0, regardless of the signs of the input operands, is stored in Aa and Aa+1.

(8) If the absolute value of the dividend mantissa is less than the absolute value of the divisor mantissa, the instruction execution time is extended by .25 microseconds.

(9) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

(10) Neither of the A registers used to store the quotient should be used as the index register specified by the x field of the next sequential instruction.

6.5.9. Load and Unpack Floating

LUF
76,04
0.75 μs alternate
1.50 μs same

$|(U)|_{34-27} \rightarrow A_{7-0}$, zerofill;

$(U)_{26-0} \rightarrow A+1_{26-0}$, signfill

The single-precision floating-point number from location U is transferred to the arithmetic section and unpacked. The absolute value of the biased characteristic of the input operand is transferred to bits 7 through 0 of Aa; bits 35 through 8 of the Aa are filled with 0 bits. The mantissa of the input operand is transferred to bits 26 through 0 of Aa+1; bits 35 through 27 of Aa+1 are filled with bits identical to the sign of the floating-point number in U.

(1) No attempt is made to normalize the input operand.

(2) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.5.10. Double Load and Unpack Floating

DFU
76,14
1.50 μs alternate
2.25 μs same

$|(U)|_{34-24} \rightarrow A_{10-0}$, zerofull; $(U)_{23-0} \rightarrow A+1_{23-0}$, signfill; $(U+1) \rightarrow A+2$

The double-precision floating-point number from locations U and U+1 is transferred to the arithmetic section and unpacked. The absolute value of the biased characteristic of the input operand is transferred to bits 10 through 0 of Aa; bits 35 through 11 of Aa are filled with 0 bits. The leftmost 24 bits of the mantissa, $(U)_{23-0}$, are transferred to bits 23 through 0 of Aa+1; bits 35 through 24 of Aa+1 are filled with bits identical to the sign of the floating-point number in locations U and U+1. The rightmost 36 bits of the mantissas (U+1) are transferred to Aa+2.

(1) None of the three A registers used to store results for this instruction should be used as the index register specified by the x field in the next sequential instruction.

(2) No attempt is made to normalize the input operand.

(3) If a = 16_8, Aa+2 is the control register at address 34_8 or 174_8. If a = 17_8, Aa+1 and Aa+2 are the control registers at addresses 34_8 and 35_8, or 174_8 and 175_8 respectively.

6.5.11. Load and Convert To Floating

LCF
76,05
1.125 μs alternate
1.875 μs same

$(U)_{35} \rightarrow A{+}1_{35}$; [normalized $(U)]_{26-0} \rightarrow A{+}1_{26-0}$;

if $(U)_{35} = 0$: $(A)_{7-0} \pm$ normalizing count $\rightarrow A{+}1_{34-27}$;

if $(U)_{35} = 1$: ones complement of $[(A)_{7-0} \pm$ normalizing count$] \rightarrow A{+}1_{34-27}$

The fixed-point number from location U is sent to the arithmetic section where it is shifted right or left, as required, to normalize it. The normalizing shift count is added to the characteristic from the rightmost eight bits of Aa if a normalizing right shift is required. It is subtracted from the characteristic if a normalizing left shift is required. The adjusted characteristic (complemented if U is negative) is packed with the normalized value from U to form a single-precision floating-point number. The packed result is stored in Aa+1. The contents of Aa remain unchanged.

(1) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt may occur (see 4.4.3).

(2) The 28 leftmost bits from Aa are ignored by the arithmetic section; $(Aa)_{7-0}$ must be prebiased.

(3) If location U contains ±0 and D8 of PSR = 0, the result stored is +0.

(4) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.5.12. Double Load and Convert To Floating

DFP
76,15
2.125 μs alternate
2.875 μs same

$(U)_{35} \rightarrow A{+}1_{35}$; [normalized $(U,U{+}1)]_{59-0} \rightarrow A{+}1_{23-0}$ and A+2;

if $(U)_{35} = 0$: $(A)_{10-0} \pm$ normalizing count $\rightarrow A{+}1_{34-24}$;

if $(U)_{35} = 1$: ones complement of $[(A)_{10-0} \pm$ normalizing count$] \rightarrow A{+}1_{34-24}$

The double-precision fixed-point number from locations U and U+1 is sent to the arithmetic section where it is shifted right or left, if necessary, to normalize it. The normalizing shift count is added to the characteristic from the rightmost 11 bits of Aa if a normalizing right shift is required. It is subtracted from the characteristic if a normalizing left shift is required. The adjusted characteristic (complemented if U is negative) is packed with the normalized value from U and U+1 to form a double-precision floating-point number and the packed result in Aa+1 and Aa+2. The contents of Aa remain unchanged.

(1) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt may occur (see 4.4.3).

(2) Neither of the A registers used to store the result should be used as an index register by the next sequential instruction.

(3) If the 72-bit input operand from U and U+1 is ±0, the result stored in +0 regardless of the sign of the 72-bit input operand.

(4) If the adjusted characteristic represents a negative number and D5 of PSR = 1, a Floating-Point Characteristic Underflow Fault Interrupt does not occur. Instead +0, regardless of the sign of the 72-bit input operand, is stored.

(5) If a = 16_8, Aa+2 is the control register at address 34_8 or 174_8. If a = 17_8, Aa+1 and Aa+2 are the control registers at addresses 34_8 and 35_8, or 174_8 and 175_8, respectively.

(6) The 25 leftmost bits from Aa are ignored by the arithmetic section; $(Aa)_{10-0}$ must be prebiased.

6.5.13. Floating Expand and Load

FEL
76,16
1.00 μs alternate
1.75 μs same 1.75

If $(U)_{35} = 0$: $(U)_{35-27} + 1600_8 \rightarrow A_{35-24}$;

if $(U)_{35} = 1$: $(U)_{35-27} - 1600_8 \rightarrow A_{35-24}$;

$(U)_{26-3} \rightarrow A_{23-0}$;

$(U)_{2-0} \rightarrow A+1_{35-33}$;

$(U)_{35} \rightarrow A+1_{32-0}$

The single-precision floating-point input operand from location U is transferred to the arithmetic section. The three fields of the operand are expanded to form a double-precision floating-point number in a 72-bit nonaddressable register, as follows:

(a) The sign bit is stored in bits 71 and 32 through 0.

(b) The 8-bit characteristic which includes a bias of 200_8 is modified to an 11-bit characteristic which includes a bias of 2000_8 and it is stored in bits 70 through 60.

(c) The 27-bit mantissa is stored in bits 59 through 33.

The contents of the left and right halves of the 72-bit register are transferred to Aa and Aa+1, respectively.

(1) If the input operand is not in the normalized single-precision floating-point format, the results stored are undefined. The following notes apply only if the input operand is a normalized number.

(2) If the mantissa of the input operand is ±0, the result stored in Aa and Aa+1 is +0 regardless of the sign of the input operand.

(3) Neither of the A registers used to store the result should be used as the index register specified by the x field of the next sequential instruction.

(4) A Floating-Point Characteristic Overflow/Underflow Fault Interrupt will not occur as a result of this instruction (see 4.4.3).

(5) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.5.14. Floating Compress and Load

FCL
76,17
1.625 μs alternate
2.375 μs same

If $(U)_{35} = 0$: $(U)_{35-24} - 1600_8 \rightarrow A_{35-27}$;

if $(U)_{35} = 1$: $(U)_{35-24} + 1600_8 \rightarrow A_{35-27}$;

$(U)_{23-0} \rightarrow A_{26-3}$;

$(U+1)_{35-33} \rightarrow A_{2-0}$

The double-precision floating-point operand from locations U and U+1 is transferred to the arithmetic section. The three fields of the input operand are compressed to form a single-precision floating-point number in a 36-bit nonaddressable register, as follows:

(a) The sign bit is stored in bit 35.

(b) The 11-bit characteristic which includes a bias of 2000_8 is modified to an 8-bit characteristic which includes a bias of 200_8 and it is stored in bits 34 through 27.

(c) The 27 leftmost bits of the mantissa (bits 23 through 0 from location U and bits 35 through 33 from location U+1) are stored in bits 26 through 0.

The content of the 36-bit register is transferred to Aa.

(1) If the input operand is not in the normalized double-precision floating-point format, the result stored is undefined. The following notes apply only if the input operand is a normalized number.

(2) A Floating-Point Characteristic Overflow Fault Interrupt occurs if the characteristic of the input operand is greater than 2177_8. A Floating-Point Characteristic Underflow Fault Interrupt occurs if the characteristic of the input operand is less than 1600_8.

(3) The contents of $U+1_{32-0}$ are ignored.

(4) If the signed mantissa is ±0, the result stored is +0 regardless of the sign and characteristic of the input operand.

(5) The A register specified by this instruction should not be used as the index register specified by the x field of the next sequential instruction.

6.5.15. Magnitude of Characteristic Difference to Upper

MCDU
76,06
0.75 μs alternate
1.50 μs same

$$\left| |(A)|_{35-27} - |(U)|_{35-27} \right| \rightarrow A+1_{8-0};\ \text{zeros} \rightarrow A+1_{35-9}$$

The characteristic of the single-precision floating-point number from location U is subtracted from the character of the single-precision floating-point number from Aa using the characteristic adder. The absolute value of the 9-bit difference is stored in bits 8 through 0 of Aa+1. Bits 35 through 9 of Aa+1 are zerofilled. The contents of Aa are not changed.

(1) The mantissas from location U and from Aa are ignored.

6.5.16. Characteristic Difference to Upper

CDU
76,07
0.75 μs alternate
1.50 μs same

$$|(A)|_{35-27} - |(U)|_{35-27} \rightarrow A+1_{8-0};\ \text{sign bits to } A+1_{35-9}$$

The characteristic of the single-precision floating-point number from location U is subtracted from the characteristic of the single-precision floating-point number from Aa using the characteristic adder. The 9-bit signed difference is stored in bits 8 through 0 of Aa+1. Bits 35 through 9 of the Aa+1 are filled with bits identical to the sign of the difference. The contents of Aa are not changed.

(1) The mantissas from location U and from Aa are changed.

6.6. SEARCH AND MASKED SEARCH INSTRUCTIONS

There are six search instructions, each of which compares the contents of either one or two A registers with the contents of main storage locations or control registers. There are eight masked search instructions, each of which compares predefined bit positions of the contents of either one or two A registers with the contents of the corresponding bit positions of main storage locations or control registers.

These are all multistage instructions. The various stages required to perform these instructions are as follows:

- An initial stage
- Repeated test stages (any number from 0 to 262,143)
- Termination stage

If indirect addressing is specified, it proceeds, as explained in 5.2.7, prior to initiation of the first test stage.

The initial stage prepares the control section and the arithmetic section for the test stages. The following steps are performed during the initial stage:

- The P register is incremented: (P) + 1 → P
- The contents of the Repeat Count Register (R^1) are transferred to the index subsection.
- The contents of the specified A registers are transferred to the arithmetic section.
- The contents of the Mask Register (R^2) are transferred to the arithmetic section for a masked search instruction.

These steps are performed only during the initial stage and are not repeated during the test stages.

The rightmost 18-bit positions of R1 contain the repeat count; that is, the maximum number of test stages to be performed. R1 must be loaded with the desired repeat count prior to initiating a search or masked search instruction; it is recommended that the leftmost 18-bit positions of R1 contain 0 bits. If the initial repeat count is zero, the termination stage is initiated immediately following the completion of the initial stage; there are no test stages. If the repeat count is not zero, a series of one or more test stages is initiated.

During each test stage, the value U is formed in the index subsection. For the search instructions, an input operand is transferred to the arithmetic section under j field control. The inputs to the test process are the values obtained using the effective U address and the A register or registers specified by the instruction.

For the masked search instructions, the content of the j field is a minor function code. The inputs to the test process are:

- the logical product of the mask from R2 and the input operand addressed by U; and
- the logical products of the mask and the specified A registers.

Each bit of the logical product is the logical product of the contents of corresponding bit positions of the two words. The logical product of two bits (X AND Y) is defined as follows:

X	Y	Logical Product (X AND Y)
0	0	0
0	1	0
1	0	0
1	1	1

The search and masked search instructions include algebraic and alphanumeric comparisons, During an algebraic comparison, the leftmost bit of each of the 36-bit values is considered to be a sign bit; a positive number is always recognized as being greater than a negative number. During an alphanumeric comparison, the leftmost bit of each of the 36-bit values is considered to be a numeric bit rather than a sign bit.

If the test process shows that the specified conditions are met, the repeat count is decremented by one and the termination stage is initiated. If the specified conditions are not met, the repeat count is decremented by one and examined. If the decremented repeat count is zero, the termination stage is initiated. If the decremented repeat count is not zero, another test stage is normally initiated. It should be noted that if x = 0 or if h = 0, the same value for U will be formed in each test stage.

As previously indicated, the termination stage is initiated if the initial repeat count is zero, if the repeat count is decremented from one to zero during the test stage, or if the conditions specified by the search or masked search instruction are detected during a test stage. If an I/O interrupt is detected during either an initial stage or one of the test stages, the termination stage is initiated immediately following that stage. The P register is reset and the repeat count is stored so that the search instruction can be resumed when the I/O interrupt condition has been satisfied.

The termination stage is used to transfer the current repeat count from the index subsection to the rightmost 18-bit positions of R1. The contents of the P register may or may not be changed during the termination stage, as follows:

- If the termination stage is entered as a result of detecting that the initial repeat count is zero during the initial stage or detecting that the decremented repeat count is zero during a test stage for which the specified conditions are not detected (no find), the contents of the P register are not changed during the termination stage. The P register contains the address of the instruction following the search or masked search instruction or the address of the instruction following the Execute instruction which referenced the search or masked search instruction (next instruction condition).

- If the termination stage is entered as a result of detecting the specified conditions during a test stage (a find has been made), the P register also incremented during the termination stage: (P) + 1 → P. Since the P register was also incremented during the initial stage, it now contains the address of the search or masked search instruction (or the address of the Execute instruction which referenced it), +2 (skip next instruction condition).

- If the termination stage is entered as a result of recognizing an I/O interrupt, the P register is decremented by 1 during the termination stage: (P) – 1 → P. This decrementation offsets the incrementation of the P register performed during the initial stage; the P register now contains the address of the search or masked search instruction, or the address of the Execute instruction which referenced it.

This address can be preserved so that when the interrupt condition has been satisfied, the search or masked search can be resumed at the point where it was terminated for the interrupt. If the search or masked search instruction is entered by means of an Execute instruction, the h field of the Execute instruction should be zero; that is, (no incrementation) so that when the program returns to the Execute instruction after an interrupt, the effective U address will again lead to the search or masked search instruction.

If the search or masked search instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search or masked search after an interrupt.

When a search or masked search is resumed after an interrupt, the initial stage is again performed to prepare the control section for the remaining test stages and to transfer the contents of the specified A register to the arithmetic section for the comparisons performed in the test stages. When h = 1 (that is, index register incrementation is specified), if the a and x fields reference the same control register, the contents of that register will have been altered by the index incrementation which occurred before the search or masked search was interrupted. As a result, when the search or masked search is resumed, the value referenced by the a field to be used in the test stages is no longer the original test value used before the interrupt occurred. Therefore when h = 1, the a field and x field should not specify the same control register so that the search or masked search instruction can be resumed in the event of an interrupt.

A word consisting of all 0 bits represents +0 and a word consisting of all 1 bits represents −0.

6.6.1. Search Equal

SE
63
2.25 + 0.75K μs
(K = number of times test is performed)

Skip NI if (U) = (A), else repeat

During the initial stage, the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa are transferred to the arithmetic section, and the P register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section under j field control. U is compared to Aa and:

- if U = Aa, the termination stage is initiated. This stage stores the remnant repeat count and increments the P register (skip to NI).
- if U ≠ Aa and the repeat count is not zero, another test stage is initiated.
- if U ≠ Aa and the repeat count is zero, the termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count must be loaded into R1 by an instruction preceding the search instruction.

(2) +0 is not equal to −0.

(3) If the x field is not zero, h = 1, and $X_i \neq \pm 0$, a different effective U address is referenced for each test stage.

(4) Normally, h = 1 and x ≠ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during the execution of the instruction.

(5) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after an interrupt.

(6) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

6.6.2. Search Not Equal

SNE
63
2.25 + 0.75K μs
(K = number of times the test is performed)

Skip NI if (U) ≠ (A), else repeat

During the initial stage, the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa are transferred to the arithmetic section, and the P register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section under j field control. U is compared to Aa and:

- if U ≠ Aa, the termination stage is initiated. The termination stage stores the remnant repeat count and increments the P register (skip NI).
- if U = Aa and the repeat count is not zero, another test stage is initiated.
- if U=Aa and the repeat count is zero, the termination stage is initiated. The termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count must be loaded into R1 by an instruction preceding the search instruction.

(2) +0 is not equal to −0.

(3) If the x field is not zero, h = 1, and $X_i \neq \pm 0$, a different effective U address is referenced for each test stage.

(4) Normally, h = 1 and x ≠ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during execution of the instruction.

(5) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after an interrupt.

(6) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

6.6.3. Search Less Than or Equal (SLE) – Search Not Greater (SNG)

SLE,SNG
64
2.25 + 0.75K μs
(K=number of times the test is performed)

Skip NI if (U)≤(A), else repeat

During the initial stage, the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa are transferred to the arithmetic section, and the P register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section under j field control. U is compared to Aa and:

- if U ≤ Aa, the termination stage is initiated. The termination stage stores the remnant repeat count and increments the P register (skip NI).
- if U > Aa and the repeat count is not zero, another test stage is initiated.
- if U > Aa and the repeat count is zero, the termination stage is initiated. The termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count must be loaded into R1 by an instruction preceding the search instruction.

(2) +0 is greater than −0.

(3) If the x field is not zero, h = 1, $X_i \neq \pm 0$, a different effective U address is referenced for each test stage.

(4) Normally, h = 1 and x ≠ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during execution of the instruction.

(5) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after an interrupt.

(6) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

6.6.4. Search Greater

SG
65
2.25 + 0.75K μs
(K=number of times the test is performed)

Skip NI if (U) > (A), else repeat

During the initial stage, the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa are transferred to the arithmetic section, and the P register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section under j field control. U is compared to Aa and:

- if U > Aa, the termination stage is initiated. The termination stage stores the remnant repeat count and increments the P register (skip NI).
- if U $\leq$ Aa and the repeat count is not zero, another test stage is initiated.
- if U $\leq$ Aa and the repeat count is zero, the termination stage is initiated. The termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count must be loaded into R1 by the instruction preceding the search instruction.

(2) +0 is greater than −0.

(3) If the x field is not zero, h = 1, and $X_i \neq \pm 0$, a different effective U address is referenced for each test stage.

(4) Normally, h = 1 and x $\neq$ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during execution of the instruction.

(5) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after an interrupt.

(6) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

6.6.5. Search Within Range

SW
66
2.25 + 0.75K μs
(K=number of times the test is performed)

Skip NI if (A) < (U) ≤ (A+1), else repeat

During the initial stage the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa and Aa+1 are transferred to the arithmetic section, and the P register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section under j field control. U is compared to Aa and Aa+1, and:

- if U > AA and U ≤ Aa + 1, the termination stage is initiated. The termination stage stores the remnant repeat count and increments the P register (skip NI).
- if U ≤ A or U > Aa+1, and the repeat count is not zero, another test stage is initiated.
- if U ≤ Aa or U > Aa + 1, and the repeat count is zero, the termination stage is initiated. The termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count must be loaded into R1 by an instruction preceding the search instruction.

(2) +0 is greater than −0.

(3) Normally, (Aa) < (Aa+1). If, however, (Aa) ≥ (Aa+1), there is no value from U which can satisfy the conditions (Aa) < (U) ≤ (Aa+1).

(4) If the x field is not zero, h = 1, and $X_i \neq \pm 0$, a different effective U address is referenced for each test stage.

(5) Normally, h = 1 and x ≠ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during execution of the instruction.

(6) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after an interrupt.

(7) If a = 17_8, Aa+1 is the control register at addresses 34_8 or 174_8 (see Table 3–6).

(8) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

6.6.6. Search Not Within Range

SNW
67
2.25 + 0.75K μs
(K=number of times the test is performed)

Skip NI if (U) $\leq$ (A) or (U) > (A+1), else repeat

During the initial stage, the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa and Aa+1 are transferred to the arithmetic section, and the P register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section under j field control. U is compared to Aa and Aa+1, and:

- if U $\leq$ Aa or U > Aa+1, the termination stage is initiated. The termination stage stores the remnant repeat count and increments the P register (skip NI).
- if U > Aa and U $\leq$ Aa+1, and the repeat count is not zero, another test stage is initiated.
- if U > Aa and U $\leq$ Aa+1, and the repeat count is zero, the termination stage is initiated. The termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count must be loaded into R1 by an instruction preceding the search instruction.

(2) +0 is greater than −0.

(3) Normally, (Aa) < (Aa+1). If, however, (Aa) $\geq$ (Aa+1), there is no value from U which can satisfy the conditions (U) $\leq$ (Aa) or (U) > (Aa+1).

(4) If the x field is not zero, h = 1, and $X_i \neq \pm 0$, a different effective U address is referenced for each test stage.

(5) Normally, h = 1 and x $\neq$ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during execution of the instruction.

(6) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after an interrupt.

(7) If a = 17_8, Aa+1 is the control register at addresses 34_8 or 174_8.

(8) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

6.6.7. Mask Search Equal

MSE
71,00
2.25 + 0.75K μs
(K=number of times the test is performed)

Skip NI if (U) AND (R2) = (A) AND (R2), else repeat

During the initial stage, the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa and R2 are transferred to the arithmetic section, the logical products of the values from Aa and R2 are formed, and the P register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section. U AND R2 is compared to Aa AND R2 and:

- if U AND R2 = Aa AND R2, the termination stage is initiated. This stage stores the remnant repeat count and increments the P register (skip NI).
- if U AND R2 $\neq$ Aa AND R2 and the repeat count is not zero, another test stage is initiated.
- if U AND R2 $\neq$ Aa AND R2 and the repeat count is zero, the termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count and mask word must be loaded into R1 and the Mask Register by instructions preceding the search instruction.

(2) +0 is not equal to −0.

(3) If the x field is not zero, h = 1, and $X_i \neq \pm 0$, a different effective U address is referenced for each test stage.

(4) Normally, h = 1 and x $\neq$ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during execution of the instruction.

(5) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after the interrupt.

6.6.8. Mask Search Not Equal

MSNE
71,01
2.25 + 0.75K μs
(K=number of times the test is performed)

Skip NI if (U) AND (R2) $\neq$ (A) AND (R2), else repeat.

During the initial stage, the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa and R2 are transferred to the arithmetic section, the logical products of the values from Aa and R2 are formed, and the 0 register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section. U AND R2 is compared to Aa AND R2 and:

- if U AND R2 $\neq$ Aa AND R2, the termination stage is initiated. This stage stores the remnant repeat count and increments the P register (skip NI).
- if U AND R2 = Aa AND R2 and the repeat count is not zero, another test stage is initiated.
- if U AND R2 = Aa AND R2 and the repeat count is zero, the termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count and mask word must be loaded into R1 and the Mask Register by instructions preceding the search instruction.

(2) +0 is not equal to −0.

(3) If the x field is not zero, h = 1, and $X_i \neq \pm 0$, a different effective U address is referenced for each test stage.

(4) Normally, h = 1 and x $\neq$ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during execution of the instruction.

(5) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after the interrupt.

6.6.9. Mask Search Less Than or Equal (MSLE) – Mask Search Not Greater (MSNG)

MSLE,MSNG
71,02
2.25 + 0.75K μs
(K=number of times the test is performed)

Skip NI if (U) AND (R2) $\leq$ (A) AND (R2), else repeat

During the initial stage, the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa and R2 are transferred to the arithmetic section, the logical products of the values from Aa and R2 are formed, and the P register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section. U AND R2 is compared to Aa AND R2 and:

- if U AND R2 $\leq$ Aa AND R2, the termination stage is initiated. This stage stores the remnant repeat count and increments the P register (skip NI).
- if U AND R2 > Aa AND R2 and the repeat count is not zero, another test stage is initiated.
- if U AND R2 > Aa AND R2 and the repeat count is zero, the termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count and mask word must be loaded into R1 and the Mask Register by instructions preceding the search instruction.

(2) +0 is greater than –0.

(3) If the x field is not zero, h = 1, and $X_i \neq \pm 0$, a different effective U address is referenced for each test stage.

(4) Normally, h = 1 and x $\neq$ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during execution of the instruction.

(5) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after the interrupt.

6.6.10. Mask Search Greater

MSG
71,03
2.25 + 0.75K μs
(K=number of times the test is performed)

Skip NI if (U) AND (R2) > (A) AND (R2), else repeat

During the initial stage, the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa and R2 are transferred to the arithmetic section, the logical products of the values from Aa and R2 are formed, and the P register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section. U AND R2 is compared to Aa AND R2 and:

- if U AND R2 > Aa AND R2, the termination stage is initiated. This stage stores the remnant repeat count and increments the P register (skip NI).
- if U AND R2 $\leq$ Aa AND R2 and the repeat count is not zero, another test stage is initiated.
- if U AND R2 $\leq$ Aa AND R2 and the repeat count is zero, the termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count and mask word must be loaded into R1 and the Mask Register by instructions preceding the search instruction.

(2) +0 is greater than −0.

(3) If the x field is not zero, h = 1, and $X_i \neq \pm 0$, a different effective U address is referenced for each test stage.

(4) Normally, h = 1 and x $\neq$ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during execution of the instruction.

(5) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after the interrupt.

6.6.11. Masked Search Within Range

MSW
71,04
2.25 + 0.75K μs
(K=number of times the test is performed)

Skip NI if (A) AND (R2) < (U) AND (R2) $\leq$ (A+1) AND (R2), else repeat.

During the initial stage, the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa, Aa+1, and R2 are transferred to the arithmetic section, the logical products of the values from Aa and R2 and the values from Aa+1 and R2 are formed, and the P register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section. The logical products are compared and:

- if (U) AND (R2) < (Aa) AND (R2) and (U) AND (R2) $\leq$ (Aa+1) AND (R2) the termination stage is initiated. This stage stores the remnant repeat count and increments the P register (skip NI).
- if (U) AND (R2) $\leq$ (Aa) AND (R2) or (U) AND (R2) > (Aa+1) AND (R2) and the repeat count is not zero, another test stage is initiated.
- if (U) AND (R2) $\leq$ (Aa) AND (R2) or (U) AND (R2) > (Aa+1) AND (R2) and the repeat count is zero, the termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count and mask word must be loaded into R1 and the Mask Register by instructions preceding the search instruction.

(2) +0 is greater than −0.

(3) Normally, (Aa) AND (R2) < (Aa+1) AND (R2). If, however, (Aa) AND (R2) $\geq$ (Aa+1) AND (R2), every possible value of U will satisfy at least one of the following conditions:

(U) AND (R2) $\leq$ (Aa) AND (R2)

or

(U) AND (R2) > (Aa+1) AND (R2)

(4) If the x field is not zero, h = 1, and $X_i \neq \pm 0$, a different effective U address is referenced for each test stage.

(5) Normally, h = 1 and x $\neq$ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during execution of the instruction.

(6) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after the interrupt.

(7) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.6.12. Masked Search Not Within Range

MSNW
71,05
2.25 + 0.75K μs
(K=number of times the test is performed)

Skip NI if (U) AND (R2) $\leq$ (A) AND (R2) or (U) AND (R2) $>$ (A+1) AND (R2), else repeat

During the initial stage, the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa and R2 are transferred to the arithmetic section, the logical products of the values from Aa and R2 and the values from Aa+1 and R2 are formed, and the P register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section. The logical products are compared and:

- if (U) AND (R2) $\leq$ (Aa) AND (R2) or (U) AND (R2) $>$ (Aa+1) AND (R2) the termination stage is initiated. This stage stores the remnant repeat count and increments the P register (skip NI).
- if (U) AND (R2) $>$ (Aa) AND (R2) and (U) AND (R2) $\leq$ (Aa+1) AND (R2) and the repeat count is not zero, another test stage is initiated.
- if (U) AND (R2) $>$ (Aa) AND (R2) and (U) AND (R2) $\leq$ (Aa+1) AND (R2) and the repeat count is zero, the termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count and mask word must be loaded into R1 and Mask Register by instructions preceding the search instruction.

(2) +0 is greater than −0.

(3) Normally, (Aa) AND (R2) $<$ (Aa+1) AND (R2). If however, (Aa) AND (R2) $\geq$ (Aa+1) AND (R2), every possible value of U will satisfy at least one of the following conditions:

(U) AND (R2) $\leq$ (Aa) AND (R2)
or
(U) AND (R2) $>$ (Aa+1) AND (R2)

(4) If the x field is not zero, h = 1, and $X_i \neq \pm 0$, a different effective U is referenced for each test stage.

(5) Normally, h = 1 and x $\neq$ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during execution of the instruction.

(6) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after the interrupt.

(7) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.6.13. Masked Alphanumeric Search Less Than or Equal

MASL
71,06
2.25 + 0.75K μs
(K=number of times the test is performed)

Skip NI if (U) AND (R2) $\leq$ (A) AND (R2), else repeat

During the initial stage, the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa and R2 are transferred to the arithmetic section, the logical products of the values from Aa and R2 are formed, and the P register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section. U AND R2 is compared alphanumerically to Aa AND R2, and:

- if U AND R2 $\leq$ Aa AND R2, the termination stage is initiated. This stage stores the remnant repeat count and increments the P register (skip NI).
- if U AND R2 > Aa AND R2 and the repeat count is not zero, another test stage is initiated.
- if U AND R2 > Aa AND R2 and the repeat count is zero, the termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count and mask word must be loaded into R1 and Mask Register by the instruction preceding the search instruction.

(2) -0 is greater than $+0$.

(3) If the x field is not zero, h = 1, and $X_i \neq \pm 0$, a different effective U address address is referenced for each test stage.

(4) Normally, h = 1 and x $\neq$ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during execution of the instruction.

(5) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after the interrupt.

6.6.14. Masked Alphanumeric Search Greater

MASG
71,07
2.25 + 0.75K μs
(K=number of times the test is performed)

Skip NI if (U) AND (R2) > (A) AND (R2), else repeat

During the initial stage, the contents of the Repeat Count Register (R1) are transferred to the index subsection, the contents of Aa and R2 are transferred to the arithmetic section, the logical products of the value from Aa and R2 are formed, and the P register is incremented.

If the initial repeat count is zero, the next instruction (NI) is initiated.

If the initial repeat count is not zero, the first test stage is initiated. During each test stage, the repeat count is decremented and the contents of U are transferred to the arithmetic section. U AND R2 is compared alphanumerically to Aa AND R2, and:

- if U AND R2 > Aa AND R2, the termination stage is initiated. This stage stores the remnant repeat count and increments the P register (skip NI).
- if U AND R2 $\leq$ Aa AND R2 and the repeat count is not zero, another test stage is initiated.
- if U AND R2 $\leq$ Aa AND R2 and the repeat count is zero, the termination stage stores zero as the remnant repeat count and the P register is not incremented.

(1) The desired repeat count and mask word must be loaded into R1 and Mask Register by instructions preceding the search instruction.

(2) -0 is greater than $+0$.

(3) If the x field is not zero, h = 1, and $X_i \neq \pm 0$, a different effective U address is referenced for each test stage.

(4) Normally, h = 1 and x $\neq$ 0 so that incrementation occurs at each test stage and a different effective U address is referenced. When index register incrementation is specified, the a field and the x field should not reference the same control register. This ensures that the test value from the A register will not be altered by the index register incrementation in the event an interrupt occurs during execution of the instruction.

(5) If the instruction specifies indirect addressing (i field = 1), the h field should be zero to enable the program to return to the same effective U address and resume the search after the interrupt.

6.7. TEST (OR SKIP) INSTRUCTIONS

Test instructions are used to read one or more words from main storage or control registers and test for certain conditions. The result of the test is used to determine whether the instruction addressed by the incremented contents of the P register (next instruction) should be performed or skipped.

The next instruction (NI) is always read from main storage. If the decision is made to skip the NI, it is discarded, the P register is incremented a second time, and the contents of the P register are then used to address the following instruction.

The timing for each instruction depends on whether NI is skipped. The first figure given is for the skip NI case; the second figure is for the execute NI case.

Indirect addressing, indexing, and index register incrementation/decrementation operate normally.

A word consisting of all 0 bits represents +0 and a word consisting of all 1 bits represents −0.

6.7.1. Test Even Parity

TEP
44
2.00 μs skip NI/1.25 μs execute NI – alternate
2.75 μs skip NI/2.00 μs execute NI – same

Skip NI if (U) AND (A) have even parity.

The contents of U are transferred to the arithmetic section under j field control, where they are used with the contents of Aa to form a 36-bit logical product.

If (U) AND (Aa) have an even number of 1 bits, the next instruction (NI) is skipped and the instruction following NI is performed.

If (U) AND (Aa) have an odd number of 1 bits, the NI is performed.

(1) The logical product is not saved.

(2) If $j = 16_8$ or 17_8, $x = 0$, $h = i = 1$, and $u = 177777_8$, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

(3) When h = 1 and the a field and the x field specify the same control register, the value transferred to the arithmetic section from Aa is undefined.

6.7.2. Test Odd Parity

TOP
45
2.00 μs skip NI/1.25 μs execute NI – alternate
2.75 μs skip NI/2.00 μs execute NI – same

Skip NI if (A) AND (U) have odd parity.

The contents of U are transferred to the arithmetic section under j field control, where they are used with the contents of Aa to form a 36-bit logical product.

If (U) AND (Aa) have an odd number of 1 bits, the next instruction (NI) is skipped and the instruction following NI is performed.

If (U) AND (Aa) have an even number of 1 bits, the NI is performed.

(1) The logical product is not saved.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

(3) When h = 1 and the a field and the x field specify the same control register, the value transferred to the arithmetic section from Aa is undefined.

6.7.3. Test Less Than or Equal to Modifier (TLEM) – Test Not Greater Than Modifier (TNGM)

TLEM,TNGM
47
1.75 μs skip NI/1.00 μs execute NI – alternate
2.50 μs skip NI/1.75 μs execute NI – same

Skip NI if $(U)_{17-0} \leq (X_a)_{17-0}$; always $(X_a)_{17-0} + (X_a)_{35-18} \rightarrow (X_a)_{17-0}$.

The contents of U are transferred to the arithmetic section under j field control. The contents of the index register addressed by the a field (Xa) are transferred to the arithmetic section. The rightmost 18 bits of the value from U are subtracted from the rightmost 18 bits of the value from Xa (this is performed as if the leftmost 18 bits of each operand were zeros).

If $(U)_{17-0} < (Xa)_{17-0}$ (the sign of the difference is positive), the next instruction (NI) is skipped and the instruction following NI is performed.

If $(U)_{17-0} > (Xa)_{17-0}$ (the sign of the difference is negative), the NI is performed.

In either case, the leftmost 18 bits from Xa are added to the rightmost 18 bits from Xa, and the sum is stored in the rightmost 18 bit positions of Xa. The leftmost 18 bit positions of Xa are not changed.

(1) When D6 of PSR = 0, the use of a = 0 should be avoided. If D6 = a = 0, Xa is the control register at address 000_8. The contents of this location may be changed at any time as the result of an interrupt (see 8.2).

(2) +0 is less than −0.

(3) Bóth Xa_{17-0} and the value from U are considered to be 18-bit numeric values with a positive sign implied.

(4) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0.

(5) Only the rightmost 18 bits of the value from U are involved in the operation. Values of 0, 1, or 3 in the j field yield the same results. Values of 16_8 or 17_8 in the j field yield the same result.

(6) If h = 1 and a = x, the specified index register is incremented or modified only once (during instruction execution).

6.7.4. Test Zero

TZ
50
1.625 μs skip NI/0.875 μs execute NI – alternate
2.375 μs skip NI/1.625 μs execute NI – same

Skip NI if (U) = ±0.

The contents of U are transferred to the arithmetic section under j field control.

If the value transferred is ±0, the next instruction (NI) is skipped and the instruction following NI is performed.

If the value transferred is not ± 0, the NI is performed.

(1) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

6.7.5. Test Nonzero

TNZ
51
1.625 μs skip NI/0.875 μs execute NI – alternate
2.375 μs skip NI/1.625 μs execute NI – same

Skip NI if (U) ≠ ±0.

The contents of U are transferred to the arithmetic section under j field control.

If the value transferred is not ±0, the next instruction (NI) is skipped and the instruction following NI is performed.

If the value transferred is ±0, the NI is performed.

(1) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

6.7.6. Test Equal

TE
52
1.625 μs skip NI/0.875 μs execute NI – alternate
2.375 μs skip NI/1.625 μs execute NI – same

Skip NI if (U) = (A).

The contents of U are transferred to the arithmetic section under j field control. The contents of Aa are also transferred to the arithmetic section.

If U = A, the next instruction (NI) is skipped and the instruction following NI is performed.

If U ≠ A, the NI is performed.

(1) +0 is not equal to −0.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0

6.7.7. Test Not Equal

TNE
53
1.625 μs skip NI/0.875 μs execute NI – alternate
2.375 μs skip NI/1.625 μs execute NI – same

Skip NI if (U) ≠ (A).

The contents of U are transferred to the arithmetic section under j field control. The contents of Aa are also transferred to the arithmetic section.

If U ≠ A, the next instruction (NI) is skipped and the instruction following NI is performed.

If U = A, the NI is performed.

(1) +0 is not equal to −0.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

6.7.8. Test Less Than or Equal (TLE) – Test Not Greater (TNG)

TLE,TNG
54
1.625 μs skip NI/0.875 μs execute NI – alternate
2.375 μs skip NI/1.625 μs execute NI – same

Skip NI if (U) $\leq$ (A).

The contents of U are transferred to the arithmetic section under j field control. The contents of Aa are also transferred to the arithmetic section.

If U $\leq$ A, the next instruction (NI) is skipped and the instruction following NI is performed.

If U > A, the NI is performed.

(1) +0 is greater than –0.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than –0, and the U value transferred to the arithmetic section is +0.

6.7.9. Test Greater

TG
55
1.625 μs skip NI/0.875 μs execute NI – alternate
2.375 μs skip NI/1.625 μs execute NI – same

Skip NI if (U) > (A).

The contents of U are transferred to the arithmetic section under j field control. The contents of Aa are also transferred to the arithmetic section.

If U > A, the next instruction (NI) is skipped and the instruction following NI is performed.

If U $\leq$ A, the NI is performed.

(1) +0 is greater than –0.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than –0, and the U value transferred to the arithmetic section is +0.

6.7.10. Test Within Range

TW
56
1.75 μs skip NI/1.00 μs execute NI – alternate
2.50 μs skip NI/1.75 μs execute NI – same

Skip NI if (A) < (U) $\leq$ (A+1).

The contents of U are transferred to the arithmetic section under j field control. The contents of Aa and Aa+1 are also transferred to the arithmetic section.

If Aa < U $\leq$ Aa+1, the next instruction (NI) is skipped and the instruction following NI is performed.

If U $\leq$ Aa or U > Aa+1, the NI is performed.

(1) +0 is greater than −0.

(2) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

(3) Normally, (Aa) < (Aa+1). If, however, (Aa) $\geq$ (Aa+1), there is no value of U that can satisfy the condition (Aa) < (U) $\leq$ (Aa+1).

(4) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

(5) When h = 1, the value transferred to the arithmetic section from register Aa+1 is undefined if Aa+1 is also the x register being incremented; that is, for the following three combinations of x and a field values: x = 15_8 and a = 0; x = 16_8 and a = 1; x = 17_8 and a = 2.

6.7.11. Test Not Within Range

TNW
57
1.75 μs skip NI/1.00 μs execute NI – alternate
2.50 μs skip NI/1.75 μs execute NI – same

Skip NI if (U) $\leq$ (A) or (U) > (A+1).

The contents of U are transferred to the arithmetic section under j field control. The contents of Aa and Aa+1 are also transferred to the arithmetic section.

If U $\leq$ Aa or U > Aa+1, the next instruction (NI) is skipped and the instruction following NI is performed.

If U > Aa and U $\leq$ Aa+1, the NI is performed.

(1) +0 is greater than −0.

(2) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

(3) Normally, (Aa) < (Aa+1). If, however, (Aa) ≥ (Aa+1), every possible value of U will satisfy at least one of the following conditions:

(U) ≤ (Aa)

or

(U) > (Aa+1)

(4) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

(5) When h = 1, the value transferred to the arithmetic section from register Aa+1 is undefined if Aa+1 is also the x register being incremented; that is, for the following three combinations of x and a field values: x = 15_8 and a = 0; x = 16_8 and a = 1; x = 17_8 and a = 2.

6.7.12. Test Positive

TP
60
1.50 μs skip NI/0.75 μs execute NI – alternate
2.25 μs skip NI/1.50 μs execute NI – same

Skip NI if $(U)_{35} = 0$.

The contents of U are transferred to the arithmetic section under j field control.

If the sign bit (bit 35) of U is a 0 bit, the next instruction (NI) is skipped and the instruction following NI is performed.

If the sign bit is a 1 bit, the NI is performed.

(1) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

6.7.13. Test Negative

TN
61
1.50 μs skip NI/0.75 μs execute NI – alternate
2.25 μs skip NI/1.50 μs execute NI – same

Skip NI if $(U)_{35} = 1$.

The contents of U are transferred to the arithmetic section under j field control.

If the sign bit (bit 35) of U is a 1 bit, the next instruction (NI) is skipped and the instruction following NI is performed.

If the sign bit is a 0 bit, the NI is performed.

(1) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(2) If j = 16_8 or 17_8, x = 0, h = i = 1, and u = 177777_8, the 18-bit output of the index adder is +0 rather than −0, and the U value transferred to the arithmetic section is +0.

6.7.14. Double-Precision Test Equal

DTE
71,17
2.375 μs skip NI/1.625 μs execute NI – alternate
3.125 μs skip NI/2.375 μs execute NI – same

Skip NI if (U,U+1) = (A,A+1).

The contents of U,U+1, Aa, and Aa+1 are transferred to the arithmetic section. U,U+1 and Aa,Aa+1 are 72-bit operands.

If U,U+1 = Aa,Aa+1, the next instruction (NI) is skipped and the instruction following NI is performed.

If U,U+1 ≠ Aa,Aa+1, the NI is performed.

(1) +0 is not equal to −0.

(2) If a = 17_8, Aa+1 is the control register address at 34_8 or 174_8.

6.8. SHIFT INSTRUCTIONS

Each shift instruction transfers either one or two words to the arithmetic section, moves or shifts the bits of the words, and stores the shifted word or words in one or two control registers.

The following basic types of shifts are provided for both single word (36-bit input operand) and double word (two 36-bit words treated as a 72-bit input operand) operations:

- Right circular
- Left circular
- Right logical
- Left logical
- Right algebraic

– For a right circular shift, a shift count of n moves the contents of all bit positions of the register holding the input operand n bit positions to the right. Bits shifted out the right end of the register appear in the leftmost bit positions vacated by the shift.

– For a left circular shift, a shift count of n moves the contents of all bit positions of the register holding the input operand n places to the left. Bits shifted out the left end of the register appear in the rightmost bit positions vacated by the shift.

For example:

A shift count of 6 for a right circular shift applied to 765432101234_8 as the input operand produces 347654321012_8 as the result. The same result is produced using a shift count of 30 for a left circular shift.

For a single word circular shift, a shift count of 72 or 36 produces the same result as a shift count of 0 (no shift). A shift count of 37 produces the same effect as a shift count of 1, a shift count of 38 produces the same effect as a shift count of 2, and so on.

– For a right logical shift, a shift count of n moves the contents of all bit positions of the register holding the input operand n places to the right. Bits shifted out the right end of the register are lost. The leftmost bit positions vacated by the shift are zerofilled.

For example:

A shift count of 6 for a right logical shift applied to 765432101234_8 as the input operand produces 007654321012_8 as the result.

– For a left logical shift, a shift count of n moves the contents of all bit positions of the input operand register n places to the left. Bits shifted out the left end of the register are lost. The rightmost bit positions vacated by the shift are zerofilled.

For example:

A shift count of 6 for a left logical shift applied to 765432101234_8 as the input operand produces 543210123400_8 as the result.

– For an algebraic shift (right only, since no left algebraic shift is provided), a shift count of n moves the contents of all bit positions of the register holding the input operand n places to the right. Bits shifted out the right end of the register are lost. The bit positions vacated by the shift are filled with bits identical to the leftmost bit (sign bit) of the original input operand.

For example:

A shift count of 6 for an algebraic shift applied to 765432101234_8 as the input operand produces 777654321012_8 as the result.

The two load shift and count instructions are basically left circular shift instructions. The shift count is determined by the configuration of the bits of the input operand. If the two leftmost bits are not identical, the shift count is zero. If the two leftmost bits are identical, the operand is shifted left circular by the minimum amount to position the bits of the input operand so that the two leftmost bits are not identical. The shift count is the count of the number of bit positions shifted. If all bits of an input operand are identical, no amount of circular shifting will position its bits so that the two leftmost bits are not identical. In this instance, the shift count is 35 (single-word operand) or 71 (double-word operand). The shift count is stored in a control register.

For all shift instructions, except the two load shift and count instructions, the input operands are specified by one or two A registers, and the shift count is specified by bits 6 through 0 of the effective U. Indirect addressing, indexing, and index register incrementation/decrementation operate normally for all shift instructions (see 5.3.4).

The shift count can be any number between 0 and 72. If a shift count of 73 to 127 (111_8 through 177_8) is specified, the result produced is undefined. If $x \neq 0$, the value in the u field of the shift instruction and the value of X_m must be chosen so that bits 17 through 7 of U will contain 0 bits.

For the two load shift and count instructions, the effective u field specifies the input operand address just as for the other load instructions. The scaled result is loaded in the specified A register (A, A+1 for Double Load Shift And Count instruction). The number of shifts required for scaling is stored in the next consecutive register A+1 (or A+2 for Double Load Shift And Count instruction).

A word consisting of all 0 bits represents +0 and a word consisting of all 1 bits represents −0.

6.8.1. Single Shift Circular

SSC
73,00
0.75 μs

Shift (A) right circularly U places.

The contents of Aa are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The value from Aa is shifted right, circularly, by the number of bit positions specified by the shift count. The shifted value is stored in Aa.

(1) The result stored is not defined for shift counts greater than 72.

(2) If $36 \leq n \leq 72$, a shift count of n produces the same result as a shift count of n−36.

6.8.2. Double Shift Circular

DSC
73,01
0.875 μs

Shift (A,A+1) right circularly U places.

The contents of Aa and Aa+1 are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The 72-bit value from Aa and Aa+1 is shifted right, circularly, the number of bit positions specified by the shift count. The shifted value is stored in Aa and Aa+1.

(1) The result stored is not defined for shift counts greater than 72.

(2) When h = 1, the value transferred to the arithmetic section from register Aa+1 is undefined if Aa+1 is also the x register being incremented; that is, for the following three combinations of x and a field values: $x = 15_8$ and a = 0; $x = 16_8$ and a = 1; $x = 17_8$ and a = 2.

(3) If $a = 17_8$, Aa+1 is the control register at address 34_8 or 174_8.

6.8.3. Single Shift Logical

SSL
73,02
0.75 μs

Shift (A) right U places, zerofill.

The contents of Aa are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The value from Aa is right shifted the number of bit positions specified by the shift count. Bits shifted out of the rightmost bit positions are lost; the vacated leftmost bit positions are zerofilled. The shifted value is stored in Aa.

(1) The result stored is not defined for shift counts greater than 72.

(2) If $36 \leq U \leq 72$, the result stored in Aa is +0.

6.8.4. Double Shift Logical

DSL
73,03
0.875 μs

Shift (A,A+1) right U places, zerofill.

The contents of Aa and Aa+1 are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The 72-bit value from Aa and Aa+1 is right shifted the number of bit positions specified by the shift count. Bits shifted out of the rightmost bit positions are lost; the vacated leftmost bit positions are zerofilled.

(1) The result stored is not defined for shift counts greater than 72.

(2) When h = 1, the value transferred to the arithmetic section from register Aa+1 is undefined if Aa+1 is also the x register being incremented; that is, for the following three combinations of x and a field values: $x = 15_8$ and a = 0; $x = 16_8$ and a = 1; $x = 17_8$ and a = 2.

(3) If $a = 17_8$, Aa+1 is the control register at address 34_8 or 174_8.

6.8.5. Single Shift Algebraic

SSA
73,04
0.75 μs

Shift (A) right U places, signfill.

The contents of Aa are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The value from Aa is right shifted the number of bit positions specified by the shift count. Bits shifted out of the rightmost bit positions are lost; bits identical to the content of bit 35 of the initial value from Aa appear in the vacated leftmost bit positions. The shifted count is stored in Aa.

(1) The result stored is not defined for shift counts greater than 72.

(2) If $35 \leq U \leq 72$, all bits of the result stored in Aa are identical to the leftmost bit of the input operand from Aa.

6.8.6. Double Shift Algebraic

DSA
73,05
0.875 μs

Shift (A,A+1) right U places, signfill.

The contents of Aa and Aa+1 are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The 72-bit value from Aa and Aa+1 is right shifted the number of bit positions specified by the shift count. Bits shifted out of the rightmost bit positions are lost; bits identical to the contents of bit 35 of the initial value from Aa appear in the vacated leftmost bit positions. The shifted value is stored in Aa and Aa+1.

(1) The result stored is not defined for shift counts greater than 72.

(2) When h = 1, the value transferred to the arithmetic section from register Aa+1 is undefined if Aa+1 is also the x register being incremented; that is, for the following three combinations of x and a field values: $x = 15_8$ and $a = 0$; $x = 16_8$ and $a = 1$; $x = 17_8$ and $a = 2$.

(3) If $a = 17_8$, Aa+1 is the control register at address 34_8 or 174_8.

6.8.7. Load Shift and Count

LSC
73,06
1.125 μs – alternate
1.875 μs – same

(U) → A; shift (A) left circularly until $(A)_{35} \neq (A)_{34}$; number of shifts → A+1.

The contents of location U are transferred to a nonaddressable 36-bit register in the arithmetic section and then shifted left, circularly, the minimum number of bit positions which will make bit 35 unequal to bit 34. The resultant scaled number is transferred to Aa and the shift count to Aa+1.

(1) If bit 35 of the value from location U is not equal to bit 34, the number is already scaled and no shift occurs: (U) → Aa; + 0 → Aa+1.

(2) If the value from location U is ±0: (U) → Aa, the shift count is 35, and 43_8 → Aa+1.

(3) If $a = 17_8$, Aa+1 is the control register at address 34_8 or 174_8.

6.8.8. Double Load Shift and Count

DLSC
73,07
2.125 μs – alternate
2.875 μs – same

(U,U+1) → A,A+1; shift (A,A+1) left circularly until $(A,A+1)_{71} \neq (A,A+1)_{70}$; number of shifts → A+2.

The contents of U and U+1 are transferred to a nonaddressable 72-bit register in the arithmetic section and then shifted left, circularly, the minimum number of bit positions which will make bit 71 unequal to bit 70. The resultant scaled number is transferred to Aa and Aa+1 and the shift count to Aa+2.

(1) If bit 71 of the value from U and U+1 is not equal to bit 70, the double length number is already scaled and no shift occurs: (U) → Aa; (U+1) → Aa+1; +0 → Aa+2.

(2) If the double length value from locations U and U+1 is ±0: (U) → Aa; (U+1) → Aa+1; the shift count is 71; and 107_8 → Aa+2.

(3) Control register Aa+2 used to store the shift count should not be used as the index register addressed by the x field in the next instruction.

(4) If a = 16_8, Aa+2 is the control register at address 34_8 or 174_8. If a = 17_8, Aa+1 and Aa+2 are the control registers at addresses 34_8 and 35_8, or 174_8 and 175_8, respectively.

6.8.9. Left Single Shift Circular

LSSC
73,10
0.75 μs

Shift (A) left circularly U places.

The contents of Aa are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The value from Aa is shifted left, circularly, the number of bit positions specified by the shift count. The shifted value is stored in Aa.

(1) The result stored is undefined for shift counts greater than 72.

(2) If $36 \leq n \leq 72$, a shift count of n produces the same result as a shift count of n–36.

6.8.10. Left Double Shift Circular

LDSC
73,11
0.875 μs

Shift (A,A+1) left circularly U places.

The contents of Aa and Aa+1 are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The 72-bit value from Aa and Aa+1 is shifted left, circularly, the number of bit positions specified by the shift count. The shifted value is stored in Aa and Aa+1.

(1) The result stored is undefined for shift counts greater than 72.

(2) When h = 1, the value transferred to the arithmetic section from register Aa+1 is undefined if Aa+1 is also the x register being incremented; that is, for the following three combinations of x and a field values: x = 15_8 and a = 0; x = 16_8 and a = 1; x = 17_8 and a = 2.

(3) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.8.11. Left Single Shift Logical

LSSL
73,12
0.75 μs

Shift (A) left U places, zerofill.

The contents of Aa are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The value from Aa is left shifted the number of bit positions specified by the shift count. Bits shifted out of the leftmost bit positions are lost; the vacated rightmost bit positions are zerofilled. The shifted value is stored in Aa.

(1) The result stored is undefined for shift counts greater than 72.

(2) If $36 \leq U \leq 72$, the result stored in Aa is +0.

6.8.12. Left Double Shift Logical

LDSL
73,13
0.875 μs

Shift (A,A+1) left U places, zerofill.

The contents of Aa and Aa+1 are transferred to the arithmetic section. The shift count from bits 6 through 0 of U is transferred to the arithmetic section. The 72-bit value from Aa and Aa+1 is left shifted the number of bit positions specified by the shift count. Bits shifted out of the leftmost bit positions are lost; the vacated rightmost bit positions are zerofilled. The shifted value is stored in Aa and Aa+1.

(1) The result stored is undefined for shift counts greater than 72.

(2) When h = 1, the value transferred to the arithmetic section from register Aa+1 is undefined if Aa+1 is also the x register being incremented; that is, for the following three combinations of x and a field values: x = 15_8 and a = 0; x = 16_8 and a = 1; x = 17_8 and a = 2.

(3) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

6.9. UNCONDITIONAL JUMP INSTRUCTION

A jump is a change in the sequence in which instructions are executed. It is accomplished by placing a new value in the P register. Each unconditional jump instruction performs a unique operation in addition to the common operation of placing a new value in the P register.

The Load Modifier And Jump instruction and the Store Location And Jump instruction both save the address that was in the P register prior to placing the new value there. The value saved is the incremented value which has been developed in preparation for addressing the next instruction in the normal consecutive sequence. For both instructions, the absolute address saved is converted to a relative address (see 9.3). The Load Modifier And Jump instruction stores the relative address in the X register addressed by the a field of the instruction. The Store Location And Jump instruction stores the relative address in the storage location addressed by U, SI, or SD, as appropriate.

The Allow All I/O Interrupts And Jump instruction allows the I/O section of the CPU to recognize and react to I/O interrupt requests. It is used subsequent to a Prevent All I/O Interrupts And Jump instruction and subsequent to the recognition of, and reaction to, an I/O interrupt request.

The Prevent All I/O Interrupts And Jump instruction is also an unconditional jump instruction. It is described, however, in the section on executive system control instructions (see 6.14).

The Jump Keys instruction can be used to specify either a conditional or unconditional jump. The Halt Jump/Halt Keys And Jump instruction specifies an unconditional jump, but the halt portion is conditional. Both of these instructions are included in the section on conditional jump instructions (see 6.10).

6.9.1. Store Location and Jump

SLJ
72,01
2.125 μs

(P) – base address modifier [BI or BD] → U_{17-0}; jump to U+1

The P register is incremented. An 18-bit relative "jump from" address is formed by subtracting either BI or BD from the incremented contents of the P register. The relative address is stored in the rightmost 18 bits of the location addressed by U, SI, or SD. The P register is cleared. The 18-bit values U+1, SI+1, and SD+1 are formed in the index subsection and either SI+1 or SD+1, as appropriate, is transferred to the P register as the "jump to" address.

(1) The contents of the a field are ignored by the control section and may be used as desired.

(2) BI is subtracted from the contents of the P register to form the relative "jump from" address if SI (or SI+1) was used for the "jump to" address for the most recent jump instruction. BD is subtracted if SD (or SD+1) was used. The value in D7 of the PSR (base register suppression designator) has no effect on the value stored as the relative "jump from" address.

(3) If U < 200_8, the 18-bit relative "jump from" address is stored in the rightmost 18 bits of the control register addressed by U and the leftmost 18 bit positions of that control register are cleared to 0 bits. If U ≥ 200_8, the 18-bit relative "jump from" address is stored in the rightmost 18 bit positions of U and the upper half of U is unchanged.

(4) If the absolute "jump to" address chosen (either SI+1 or SD+1) is less than 200_8, the next instruction is taken from the main storage location addressed by the chosen value rather than from a control register.

(5) If D3 = 1 or D2 = 1 (or both), and the main storage limits set in the Storage Limits Register are violated by the absolute address of the location in which the relative "jump from" address is to be stored, a Guard Mode Fault Interrupt occurs. The relative address stored by a Load Modifier And Jump (LMJ) or Store Location And Jump (SLJ) instruction at the Guard Mode Fault Interrupt location is the "jump from" address.

(6) If D3 = 0 and D2 = 1, and the main storage limits set in the Storage Limits Register are violated by the absolute "jump to" address transferred to the P register, a Guard Mode Fault Interrupt occurs.

The relative address captured by a Load Modifier And Jump or Store Location And Jump instruction at the Guard Mode Fault Interrupt location is the relative "jump from" address.

(7) The relative "jump from" address is stored in the low order 18 bits of a word. If this 18-bit relative "jump from" address is larger than 16 bits, the two high order bits will be interpreted as h and i field bits if the address is used in an instruction. The instruction may produce erroneous results.

6.9.2. Load Modifier and Jump

LMJ
74,13
0.875 μs

(P) – base address modifier [BI or BD] → Xa_{17-0}; jump to U

The P register is incremented. An 18-bit relative "jump from" address is formed by subtracting either BI or BD from the incremented contents of the P register. The relative address is stored in the rightmost 18 bits of the index register specified by the a field. The leftmost 18 bits of that index register are not affected. The 18-bit values U, SI, and SD are formed in the index subsection and either SI or SD, as appropriate, is transferred to the P register as the "jump to" address.

(1) If D6 = 0 and the value in the a field is zero, the relative "jump from" address is stored in the control register at address 000_8. Such use should be avoided because the contents of the control register at address 000_8 are replaced by the contents of the Processor State Register (PSR) at any interrupt.

(2) If index register incrementation is specified, the relative "jump from" address is stored in the index register specified by the a field after the new value for Xm is stored in the index register specified by the x field. As a consequence, if the value in the a field is not zero and it is the same as the value in the x field, it makes no difference whether the value in the h field is a 0 bit or a 1 bit.

(3) BI is subtracted from the contents of the P register to form the relative "jump from" address if SI (or SI+1) was used for the "jump to" address for the most recent jump instruction. BD is subtracted if SD (or SD+1) was used. The value in D7 of PSR has no effect on the value stored as the relative "jump from" address.

(4) If the absolute "jump to" address chosen (either SI or SD) is less than 200_8, the next instruction is taken from the main storage location addressed by the chosen value rather than from a control register.

(5) If D3 = 0 and D2 = 1, and the main storage limits set in the Storage Limits Register are violated by the absolute "jump to" address to be transferred to the P register, the relative "jump from" address is not stored and a Guard Mode Fault Interrupt occurs. The relative address saved by a Load Modifier And Jump or Store Location And Jump instruction at the Guard Mode Fault Interrupt location is the relative "jump from" address +1.

6.9.3. Allow All I/O Interrupts and Jump

AAIJ
74,07
0.75 μs

Allow all I/O interrupts and jump to U.

The I/O section of the CPU can recognize and react to pending I/O interrupt requests or to any I/O interrupt request received following the completion of an instruction. The instruction stored in location U is executed next.

(1) The following are classed as I/O interrupts:

- All monitor interrupts
- All external interrupts including Console External Interrupt and Day Clock Interrupt
- All system I/O interrupts including Interprocessor Interrupt, Real Time Clock (RTC) Interrupt, and Power Loss Interrupt
- All I/O parity error interrupts including Access Control Word Parity Interrupt and I/O Data Parity Error Interrupt

(2) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(3) This is not a privileged instruction. When D2 of PSR = 1, an Allow All I/O Interrupts And Jump instruction can be performed without causing a Guard Mode Fault Interrupt.

(4) This instruction is used to allow I/O interrupts following the honoring of any interrupt or following execution of a Prevent All I/O Interrupts and Jump instruction (see 6.14.1).

(5) An I/O interrupt request is never honored between the execution of the Allow All I/O Interrupts And Jump instruction and the instruction jumped to. The instruction at the "jump to" address is always executed next by the CPU.

(6) The Allow All I/O Interrupts And Jump instruction enables the interrupt circuitry which is disabled by a Prevent All I/O Interrupts And Jump instruction. It is also used to enable the interrupt circuitry which is disabled during the response to an interrupt. The Allow All I/O Interrupts And Jump does not enable the interrupt circuitry which is disabled by a Prevent All Channel External Interrupts instruction. The Allow All Channel External Interrupts instruction enables this interrupt circuitry (see 6.13.12 and 6.13.13).

The Allow All I/O Interrupts And Jump instruction does not enable the Day Clock which is disabled by the Disable Day Clock instruction. The Enable Day Clock instruction performs this function (see 6.14.7 and 6.14.8).

6.10. CONDITIONAL JUMP INSTRUCTIONS

Each of the conditional jump instructions performs a test for a specific condition (or set of conditions). If the condition is satisfied, the value U is transferred to the P register and the instruction addressed by U is executed next. If the condition is not satisfied, the next instruction (NI) is executed.

For most of the conditional jump instructions, two execution times are shown: the time required when the jump is performed is listed first, followed by the time required when NI is executed.

There are three conditional jump instructions which are not included in this section. They are Jump On Input Channel Busy (see 6.13.3), Jump On Output Channel Busy (see 6.13.7), and Jump On Function In Channel (see 6.13.11).

A word consisting of all 0 bits represents +0 and a word consisting of all 1 bits represents −0.

6.10.1. Jump Greater and Decrement

JGD
70
0.75 μs jump/1.50 μs NI

Jump to U if $(\text{control register})_{ja} > 0$; go to NI if $(\text{control register})_{ja} \leq 0$; always $(\text{control register})_{ja} - 1 \rightarrow \text{control register}_{ja}$.

If the 36-bit signed number in the control register addressed by the 7-bit ja field is greater than zero (bit 35 contains a 0 bit and the number does not consist of all 0 bits), the instruction at location U is executed next. If the number is less than, or equal to, zero (bit 35 contains a 1 bit or the number consists of all 0 bits), the next instruction (NI) is executed. In either case, the number is decremented by one and the difference is stored in the control register addressed by the ja field.

(1) A Guard Mode Fault Interrupt occurs (if guard mode is set) when the ja field specifies a value in the range 40_8 through 100_8, or 120_8 through 177_8 when D2 = 1. This is true regardless of the value of D6 (User/Exec ABR).

(2) Zero should not be used in the ja field because the control register at address 000_8 is used as a temporary storage location for the contents of the PSR. The contents of this control register are destroyed whenever an interrupt occurs.

(3) The leftmost bit in the j field is ignored by the control section. It is recommended that a 0 bit be used as the leftmost bit of the j field.

6.10.2. Double-Precision Jump Zero

DJZ
71,16
1.625 μs jump/0.875 μs NI

Jump to U if (A,A+1) = ±0; go to NI if (A,A+1) ≠ ±0.

If the 72-bit operand contained in Aa and Aa+1 is ±0, the instruction at location U is executed next. If the operand is not ±0, the next instruction (NI) is executed.

(1) If a = 17_8, Aa+1 is the control register at address 34_8 or 174_8.

(2) When h = 1, the value transferred to the arithmetic section from register Aa+1 is undefined if Aa+1 is also the x register being incremented; that is, for the following three combinations of x and a field values: x = 15_8 and a = 0; x = 16_8 and a = 1; x = 17_8 and a = 2.

6.10.3. Jump Positive and Shift

JPS
72,02
1.50 μs jump/0.75 μs NI

Jump to U if $(A)_{35} = 0$; go to NI if $(A)_{35} = 1$; always shift (A) left, circularly, one bit position.

If bit 35 of Aa contains a 0 bit, the instruction at location U is executed next. If bit 35 contains a 1 bit, the next instruction (NI) is executed. The contents of Aa are always shifted left, circularly, one bit position.

(1) The bit shifted out of bit 35 of Aa is shifted to bit 0 of Aa.

6.10.4. Jump Negative and Shift

JNS
72,03
1.50 μs jump/0.75 μs NI

Jump to U if $(A)_{35} = 1$; go to NI if $(A)_{35} = 0$; always shift (A) left, circularly, one bit position.

If bit 35 of Aa is a 1 bit, the instruction at location U is executed next. If bit 35 is a 0 bit, the next instruction (NI) is executed. The contents of Aa are always shifted left, circularly, one bit position.

(1) The bit shifted out of bit 35 of Aa is shifted to bit 0 of Aa.

6.10.5. Jump Zero

JZ
74,00
1.50 μs jump/0.75 μs NI

Jump to U if (A) = ±0; go to NI if (A) ≠ ±0.

If (Aa) is ±0, the instruction at location U is executed next. If Aa does not contain ±0, the next instruction (NI) is executed.

6.10.6. Jump Nonzero

JNZ
74,01
1.50 μs jump/0.75 μs NI

Jump to U if (A) ≠ ±0; go to NI if (A) = ±0.

If (Aa) is not ±0, the instruction at location U is executed next. If (Aa) is ±0, the next instruction (NI) is executed.

6.10.7. Jump Positive

JP
74,02
1.50 μs jump/0.75 μs NI

Jump to U if $(A)_{35} = 0$; go to NI if $(A)_{35} = 1$.

If bit 35 of Aa is a 0 bit, the instruction at location U is executed next. If bit 35 is a 1 bit, the next instruction (NI) is executed.

6.10.8. Jump Negative

JN
74,03
1.50 μs jump/0.75 μs NI

Jump to U if $(A)_{35} = 1$; go to NI if $(A)_{35} = 0$.

If bit 35 of Aa is a 1 bit, the instruction at location U is executed next. If bit 35 is a 0 bit, the next instruction (NI) is executed.

6.10.9. Jump (J) – Jump Keys (JK)

J,JK
74,04
0.75 μs

Jump to U if a = 0 or if a = lit SELECT JUMPS indicator; go to NI if neither is true.

If the a field contains all 0 bits, the instruction at location U is executed next. If the a field contains a value in the range of 1 through 15 (1_8 through 17_8) and the correspondingly numbered SELECT JUMPS switch/indicator on the operator's console is lit, the instruction at location U is executed next; if the correspondingly numbered SELECT JUMPS switch/indicator is not lit, the next instruction (NI) is executed.

(1) The indicator for each of the 15 SELECT JUMPS switch/indicators is turned on by pressing that SELECT JUMPS switch/indicator. Each is turned off by pressing the associated RELEASE JUMPS switch/indicator. Either can be done while the CPU is running.

(2) Care should be exercised in using a value other than all 0 bits in the a field if the program is to run concurrently with one or more other programs or in a multiprocessor system. Any other program may include a Jump Keys instruction with the same value in the a field and specify that it is to be run with the corresponding SELECT JUMPS switch/indicator set. A set of SELECT JUMPS switch/indicators is physically associated with each CPU in a multiprocessor system. Each set affects instructions executed on the related processor only.

6.10.10. Halt Jump (HJ) – Halt Keys And Jump (HKJ)

HJ,HKJ
74,05
0.75 μs

Stop if a = 0 or if [a AND lit SELECT STOPS indicators] $\neq$ 0; on restart or continuation jump to U.

If the a field contains all 0 bits, the execution of program instruction halts. If the a field contains a 1 bit in a bit position which corresponds to a lit SELECT STOPS switch/indicator on the operator's console, the program halts. In either case, a manual restart causes the instruction at location U to be executed next.

If neither of the conditions described above is fulfilled, the instruction at location U is executed and the program does not halt.

(1) If the CPU is operating in the guard mode (D2 of PSR = 1) when a halt condition is satisfied by a Halt Keys And Jump instruction, the CPU does not halt. Instead, it proceeds immediately with the jump.

(2) The indicator for each of the four SELECT STOPS switch/indicators is turned on by pressing one of the SELECT STOPS switch/indicators when the CPU is not in guard mode. They are turned off by pressing the associated RELEASE STOPS switch.

(3) Care should be exercised in using a value other than all 0 bits in the a field if the program is to run concurrently with one or more other programs, or in a multiprocessor system. Any other program may have a Halt Keys And Jump instruction with the same value in the a field and specify it is to be run with the corresponding SELECT STOPS switch/indicator lit. In a multiprocessor system, each set of SELECT STOPS switch/indicators is physically related to a particular CPU and it affects instruction execution on the associated CPU only.

(4) When a halt occurs, the instruction from location U has been read and is in the Maintenance Panel F0 register. The P register contains the address of the following instruction (U + 1).

6.10.11. Jump No Low Bit

JNB
74,10
1.50 μs jump/0.75 μs NI

Jump to U if $(A)_0 = 0$; go to NI if $(A)_0 = 1$.

If bit 0 of Aa is a 0 bit, the instruction at location U is executed next. If bit 0 is a 1 bit, the next instruction (NI) is executed.

(1) If the Jump No Low Bit instruction is used to determine whether the value in Aa is an even or an odd integer, consideration must be given to the sign of the value.

6.10.12. Jump Low Bit

JB
74,11
1.50 μs jump/0.75 μs NI

Jump to U if $(A)_0 = 1$; go to NI if $(A)_0 = 0$.

If bit 0 of Aa is a 1 bit, the instruction at location U is executed next. If bit 0 is a 0 bit, the next instruction (NI) is executed.

(1) If a Jump Low Bit instruction is used to determine whether the value in Aa is an even or an odd integer, consideration must be given to the sign of the value.

6.10.13. Jump Modifier Greater And Increment

JMGI
74,12
1.50 μs jump/0.75 μs NI

Jump to U if $(Xa)_{17-0} > 0$; go to NI if $(Xa)_{17-0} \leq 0$; always $(Xa)_{17-0} + (Xa)_{35-18} \rightarrow Xa_{17-0}$.

If the signed number in bits 17 through 0 of the X register specified by the a field is greater than zero (bit 17 is a 0 bit and the number does not consist of all 0 bits), the instruction at location U is executed next. If the number is less than or equal to zero (bit 17 is a 1 bit or the number consists of all 0 bits), the next instruction (NI) is executed. In either case, the signed number in bits 35 through 18 of the X register is added to the signed number in bits 17 through 0 and the sum is stored in bits 17 through 0 of the X register.

(1) The number in Xa_{17-0} before the addition is tested rather than the number resulting from the addition.

(2) If a = x and h = 1, the specified index register is effectively modified only once for each execution of the instruction.

(3) In user operating mode (D6 of the PSR = 0, if the a field is zero, the control register specified is the PSR Temporary Storage register at control register address 000_8. When an interrupt occurs, the contents of the PSR are stored in this control register. Thus the original contents of this control register will be destroyed any time an interrupt occurs.

6.10.14. Jump Overflow

JO
74,14
1.50 μs jump/0.75 μs NI

Jump to U if D1 of PSR = 1; go to NI if D1 = 0.

If the overflow designator (D1) in the Processor State Register (PSR) is a 1 bit, the instruction at location U is executed next. If D1 is a 0 bit, the next instruction (NI) is executed.

(1) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(2) Executing the Jump Overflow instruction does not change D1.

(3) See 4.3.3.1 for additional information about the overflow designator.

6.10.15. Jump No Overflow

JNO
74,15
1.50 μs jump/0.75 μs NI

Jump to U if D1 of PSR = 0; go to NI if D1 = 1.

If the overflow designator (D1) in the Processor State Register (PSR) is a 0 bit, the instruction at location U is executed. If D1 is a 1 bit, the next instruction (NI) is executed.

(1) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(2) Executing the Jump No Overflow instruction does not change D1.

(3) See 4.3.3.1 for additional information about the overflow designator.

6.10.16. Jump Carry

JC
74,16
1.50 μs jump/0.75 μs NI

Jump to U if D0 of PSR=1; go to NI if D0=0.

If the carry designator (D0) in the Processor State Register (PSR) is a 1 bit, the instruction at location U is executed next. If the D0 is a 0 bit, the next instruction (NI) is executed.

(1) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(2) Performing the Jump Carry instruction does not change D0.

(3) See 4.3.3.2 for additional information about the carry designator.

6.10.17. Jump No Carry

JNC
74,17
1.50 μs jump/0.75 μs NI

Jump to U if D0 of PSR = 0; go to NI if D0 = 1.

If the carry designator (D0) in the Processor State Register (PSR) is a 0 bit, the instruction at location U is executed next. If D0 is a 1 bit, the next instruction (NI) is executed.

(1) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(2) Performing the Jump No Carry instruction does not change D0.

(3) See 4.3.3.2 for additional information about the carry designator.

6.11. LOGICAL INSTRUCTIONS

The three logical operations are the Logical Inclusive OR (referred to as the Logical OR and symbolized by OR), the Logical Exclusive OR (symbolized by XOR), and the Logical AND (symbolized by AND). Each of these instructions uses two input operands. One input operand is obtained from location U and the other from A register. Table 6-1 lists the four possible combinations of the two bits from any bit position of the two input operands and the result produced for that bit position for each of the three basic operations.

INPUT BITS		OUTPUT (RESULT) BIT		
FIRST OPERAND	SECOND OPERAND	OR	XOR	AND
0	0	0	0	0
0	1	1	1	0
1	0	1	1	0
1	1	1	0	1

Table 6-1. Truth Table for Logical OR, XOR, and AND

The Masked Load Upper instruction performs a compound logical operation; the contents of selected bit positions of one operand are merged with the contents of the remaining bit positions of a second operand.

For each of the logical instructions the result is stored in an A register.

6.11.1. Logical OR

OR
40
0.75 μs - alternate
1.50 μs - same

(A) OR (U) ⟶ A+1

The contents of Aa are transferred to the arithmetic section. The contents of U are transferred to the arithmetic section under j field control. A 36-bit result is formed in the arithmetic section, as follows:

- The result contains a 1 bit in each bit position for which the corresponding bit position of either (or both) of the input operands contains a 1 bit.
- The result contains a 0 bit in each bit position for which the corresponding bit position of both input operands contains a 0 bit.

The result is stored in Aa+1.

6.11.2. Logical Exclusive OR

XOR
41
0.75 μs - alternate
1.50 μs - same

(A) XOR (U) ⟶ A+1

The contents of Aa are transferred to the arithmetic section. The contents of U are transferred to the arithmetic section under j field control. A 36-bit result is formed in the arithmetic section, as follows:

- The result contains a 1 bit in each bit position for which the corresponding bit position of either (but not both) of the input operands contains a 1 bit.
- The result contains a 0 bit in each bit position for which the contents of the corresponding bit position of the input operands are both 0 bits or both 1 bits.

The result is stored in Aa+1.

(1) When h = 1 and the a field and the x field specify the same control register, the value transferred to the arithmetic section from Aa is undefined.

6.11.3. Logical AND

AND
42
0.75 μs - alternate
1.50 μs - same

(A) AND (U) ⟶ A+1

The contents of Aa are transferred to the arithmetic section. The contents of U are transferred to the arithmetic section under j field control. A 36-bit result is formed in the arithmetic section, as follows:

- The result contains a 1 bit in each bit position for which the corresponding bit position of both input operands contains a 1 bit.
- The result contains a 0 bit in each bit position for which the corresponding bit position of either (or both) of the input operands contains a 0 bit.

The result is stored in Aa+1

(1) When h = 1 and the a and the x fields specify the same control registers, the value transferred to the arithmetic section from Aa is undefined.

6.11.4. Masked Load Upper

MLU
43
0.75 μs - alternate
1.50 μs - same

[(U) AND (R2)] OR [(A) AND (R2)] ⟶ A+1

The contents of Aa and R2 are transferred to the arithmetic section. The contents of U are transferred to the arithmetic section under j field control. A 36-bit result is formed in the arithmetic section, as follows:

- The result contains a 1 bit in each bit position for which the corresponding bit position of the operand from U and the operand from R2 both contain 1 bits.
- The result contains a 1 bit in each bit position for which the corresponding bit position of the operand from Aa and the ones complement of the operand from R2 both contain 1 bits.
- The result contains 0 bits in the remaining bit positions.

The result is stored in Aa+1.

(1) The desired value must be loaded in R2 (Mask Register) by an instruction preceding the Masked Load Upper instruction.

6.12. MISCELLANEOUS INSTRUCTIONS

Each of the four following instructions is classed as miscellaneous.

6.12.1. Execute

EX
72,10
0.75 μs

Execute the instruction at U.

The P register is incremented provided the instruction was addressed by the contents of the P register. The instruction at location U is transferred to the control section to replace the Execute instruction as the next instruction to be performed.

(1) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(2) The remote instruction, specified by U, is always obtained from a main storage location (never from a control register). The value of U is a relative main storage address.

(3) Execute instructions may be cascaded; that is, the instruction in the remote location may be an Execute instruction.

(4) The P register is not incremented after the instruction addressed by U is transferred to the F0 register for execution.

(5) Generally, an I/O interrupt cannot occur between the time an Execute instruction is started and the instruction (or chain of instructions) it leads to has been completed except when an Execute instruction leads to a repeated instruction (see 6.3.8 and 6.6). I/O interrupt cannot occur between the start of the Execute instruction and the completion of the initial stage of the repeated instruction. The interrupt, however, can cause initiation of a termination stage immediately following completion of the initial stage or any time thereafter in order to permit the I/O interrupt to occur.

(6) If an Execute instruction leads to a repeated instruction, index register incrementation should not be specified for the Execute instructions or for any indirect addressing sequence involved (see 6.3.8, note (6) and 6.6).

6.12.2. Executive Return

ER
72,11
1.375 μs
Interrupt to 242_8

The contents of the Processor State Register (PSR) are stored in the control register at address 000_8. D7 and D6 of the PSR are loaded with 1 bits. D8, D5 through D0, and QW of the PSR are cleared to 0 bits. BI, BS, and BD of the PSR are not changed. The address of the next instruction to be performed is 242_8 + MSR rather than the incremented value in the P register.

(1) If all of the following conditions prevail,

- indirect addressing is specified (i = 1, D7 = 0);
- guard mode/storage limits protection is enabled (D3D2 = 01); and
- the value SI or SD, as applicable, violates the main storage limits set in the Storage Limits Register,

then a Guard Mode/Storage Limits Fault Interrupt will occur.

(2) If indirect addressing is not specified (i = 0 or D7 = 1 or both), the value in the μ field has no effect on the instruction or its timing.

(3) If x = 0 and h = 1, then Xm + Xi $\longrightarrow$ Xm.

(4) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(5) This is not a privileged instruction. When guard mode is specified (D2=1), it can be performed without causing a Guard Mode Fault Interrupt.

6.12.3. Test And Set

TS
73,17
1.625 μs interrupt/0.875 μs NI - alternate
2.00 μs interrupt/2.00 μs NI - same
If $(U)_{30} = 1$: interrupt to 244_8; if $(U)_{30} = 0$: go to NI; always
$01_8 \longrightarrow U_{35-30}$; $(U)_{29-0}$ unchanged.

An extended main storage cycle is initiated to read and then write at main storage location U. If bit 30 of the word read from location U is a 1 bit, the instruction at location 244_8 is executed next. If bit 30 of the word read from location U is a 0 bit, the next instruction (NI) is executed. The write portion of the extended main storage cycle includes writing 01_8 in bits 35 through 30 of main storage location U. Bits 29 through 0 at location U are not examined and are never altered.

(1) If U $\leq 177_8$, the result produced by the instruction is undefined.

(2) The P register is incremented twice as a result of performing a Test And Set instruction which leads to an interrupt. Therefore, the address saved by an LMJ or SLJ instruction at 244_8 is P + 2 rather than P + 1.

(3) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(4) In the NI case with alternate bank access, completion of the NI is delayed by .375 microsecond if the U operand addressed by the NI is in the same bank as the operand addressed by the Test And Set instruction.

(5) This is not a privileged instruction. When D2 of PSR = 1, a TS instruction can be performed without causing a Guard Mode Fault Interrupt.

6.12.4. No Operation

NOP
74,06
0.75 μs
Proceed to next instruction

This instruction ensures that there is an interval of at least 0.75 microsecond between the end of cycle 5 of the main timing chain for the instruction which precedes it and the start of cycle 1 of the main timing chain of the instruction which follows it.

(1) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(2) The only effects that the values in the x, h, i, and u fields can have on the operation is the index register incrementation obtained when x $\neq$ 0 and h = 1, and the indirect addressing delay introduced when i = 1 and D7 of PSR = 0. It is recommended that the x, h, i, and u fields contain all 0 bits.

6.13. INPUT/OUTPUT INSTRUCTIONS

The instructions discussed in this section include:

- Two instructions used to condition the CPU to accept input from a peripheral subsystem via an input channel.
- Four instructions used to condition the CPU to send output (data or function words) to a peripheral subsystem via an output channel.
- Three conditional jump instructions used to determine the status of an input or output channel.
- Two instructions used to disconnect or deactivate an input or output channel.

Each of these instructions applies to the particular I/O channel specified by the logical OR of the four-bit values in the a field of the instruction and the Channel Select Register (CSR). This is symbolized by [a OR CSR]. The logical OR operation is explained in 6.11. The CSR is loaded under program control as explained in 6.14.10.

Two I/O instructions are provided to control recognition by the I/O section of External Interrupt signals from the various peripheral subsystems: one prevents recognition, the other allows recognition.

Each I/O channel operates in Internally Specified Index (ISI) or Externally Specified Index (ESI) mode. The ESI mode is used in conjunction with communications multiplexing equipment. Certain of the I/O instructions have different effects according to the mode of the channel.

Input/output operations are explained in detail in Section 7.

All of the I/O instructions are privileged or guard mode protected instructions. An attempt to perform any I/O instruction when the Processor State Register (PSR) specifies guard mode (D2 = 1) results in a Guard Mode Fault Interrupt (see 8.3.2.2).

6.13.1. Load Input Channel

LIC
75,00
0.75 μs - alternate
1.50 μs - same

For channel [a OR CSR]: (U) ⟶ IACR; set input active; clear input monitor.

The word from location U is transferred to the Input Access Control Register (IACR) at address 40_8 + [a OR CSR]. The input active control circuit is set for the specified channel.

The input monitor control circuit is cleared for the specified channel.

(1) If D2 of PSR = 1, an attempt to perform a Load Input Channel instruction causes a Guard Mode Fault Interrupt and none of the above steps occurs.

(2) The transfer of a word to the IACR occurs only if [a OR CSR] specifies an ISI channel, never if it specifies an ESI channel.

6.13.2. Load Input Channel And Monitor

LICM
75,01
0.75 μs - alternate
1.50 μs - same

For channel [a OR CSR]: (U) ⟶ IACR; set input active; set input monitor.

The word from location U is transferred to the Input Access Control Register (IACR) at address 40_8 + [a OR CSR]. The input active and the input monitor control circuits are set for the specified channel.

(1) If D2 of PSR = 1, an attempt to perform a Load Input Channel And Monitor instruction causes a Guard Mode Fault Interrupt and none of the above steps occurs.

(2) The transfer of a word to the IACR occurs only if [a OR CSR] specifies an ISI channel, never if it specifies an ESI channel.

6.13.3. Jump On Input Channel Busy

JIC
75,02
0.75 μs

Jump to U if input active is set for channel [a OR CSR]; go to NI if input active is clear.

If the input active control circuit for channel [a OR CSR] is set, U is transferred to the P register. If the input active control circuit for the specified channel is in the cleared state, the next instruction (NI) is executed.

(1) If D2 of PSR = 1, an attempt to perform a Jump On Input Channel Busy instruction causes a Guard Mode Fault Interrupt; in this instance, the contents of the P register are not affected by the status of the input active control circuit.

6.13.4. Disconnect Input Channel

DIC
75,03
0.75 μs

For channel [a OR CSR]: clear input active; clear input monitor.

(1) If D2 of PSR = 1, an attempt to perform a Disconnect Input Channel instruction causes a Guard Mode Fault Interrupt and neither of the above steps occurs.

(2) The only effects that the values in the x, h, i, and u fields can have on the operation is the index register incrementation obtained when $x \neq 0$ and $h = 1$, and the indirect addressing delay introduced when $i = 1$ and D7 of PSR = 0. It is recommended that the x, h, i, and u fields contain all 0 bits.

(3) If the conditions which normally lead to an Input Monitor Interrupt are present on the channel specified by the DIC instruction but are not fully recognized by both the I/O section and the control section before the Disconnect Input Channel instruction is performed, the Input Monitor Interrupt does not occur.

6.13.5. Load Output Channel

LOC
75,04
0.75 μs - alternate
1.50 μs - same

For channel [a OR CSR]: (U) $\longrightarrow$ OACR; set output active; clear output monitor; clear external function (ISI only).

The word from location U is transferred to the Output Access Control Register (OACR) at address 60_8 + [a OR CSR]. The output active control circuit is set for the specified channel. The output monitor control circuit is cleared for the specified channel. For ISI only, the external function circuit on the specified channel is cleared.

(1) If D2 of PSR = 1, an attempt to perform a Load Output Channel instruction causes a Guard Mode Fault Interrupt and none of the above steps occurs.

(2) The transfer of a word to the OACR occurs only if [a OR CSR] specifies an ISI channel, never if it specifies an ESI channel.

6.13.6. Load Output Channel And Monitor

LOCM
75,05
0.75 μs
1.50 μs

For [a OR CSR]: (U) ⟶ OACR; set output active; set output monitor; clear external function (ISI only).

The word from location U is transferred to the Output Access Control Register (OACR) at address 60_8 + [a OR CSR]. The output active and output monitor control circuits are set for the specified channel. For ISI only, the external function circuit for the specified channel is cleared.

(1) If D2 of PSR = 1, an attempt to perform a Load Output Channel And Monitor instruction causes a Guard Mode Fault Interrupt and none of the above steps occurs.

(2) The transfer of a word to the OACR occurs only if [a OR CSR] specifies an ISI channel, never if it specifies an ESI channel.

6.13.7. Jump On Output Channel Busy

JOC
75,06
0.75 μs

Jump to U if output active is set for channel [a OR CSR]; go to NI if output active is clear.

If the output active control circuit for channel [a OR CSR] is set, U is transferred to the P register. If the output active control circuit for the specified channel is in the cleared state, the next instruction (NI) is executed.

(1) If D2 of PSR = 1, an attempt to perform a Jump On Output Channel Busy instruction causes a Guard Mode Fault Interrupt; in this instance, the contents of the P register are not affected by the status of the output active control circuit.

(2) This instruction tests the state of the output active control circuit for the specified channel.

- For an ISI channel: This control circuit is set when a Load Output Channel, Load Output Channel And Monitor, Load Function In Channel or Load Function In Channel And Monitor instruction specifying that channel is performed.
- For an ESI channel: this control circuit is set when a Load Output Channel or Load Output Channel And Monitor instruction specifying that channel is performed.

- For an ISI channel: this control circuit is cleared when the peripheral subsystem attached to that channel turns on the Output Data Request (ODR) signal at a time when the count in the W field of the corresponding OACR is +0.
- For an ESI channel: this control circuit is cleared when the count in the W field of the associated OACR is decremented from 1 to 0, and under certain circumstances when the count in the W field is decremented from 2 to 1 (see 7.3.3.2).

6.13.8. Disconnect Output Channel

DOC
75,07
0.75 μs

For channel [a OR CSR]: clear output active; clear output monitor; clear external function.

(1) If D2 of PSR = 1, an attempt to perform a Disconnect Output Channel instruction causes a Guard Mode Fault Interrupt and none of the above steps occurs.

(2) The only effects that the values in the x, h, i, and u fields can have on the operation is the index register incrementation obtained when $x \neq 0$ and h = 1, and the indirect addressing delay introduced when i = 1 and D7 of PSR = 0. It is recommended that the x, h, i, and u fields contain all 0 bits.

(3) If the conditions which normally lead to an Output Monitor Interrupt or a Function Monitor Interrupt are present on the channel specified by the DOC instruction but are not fully recognized by both the I/O section and the control section before the DOC instruction is performed, the Monitor Interrupt does not occur.

6.13.9. Load Function Channel

LFC
75,10
0.75 μs – alternate
1.50 μs – same

For channel [a OR CSR]: (U) → OACR; set output active (ISI only), external function, and force external function, clear output monitor (ISI only).

The word from location U is transferred to the Output Access Control Register (OACR) at address 60_8 + [a OR CSR]. The output active (ISI only), external function, and force external function control circuits are set for the specified channel. The output monitor control circuit is cleared (ISI only) for the specified channel.

(1) If D2 of PSR = 1, an attempt to perform a Load Function Channel instruction causes a Guard Mode Fault Interrupt and none of the above steps occurs.

(2) If an ESI channel is specified by a Load Function Channel instruction, the output active, external function, and output monitor control circuits are not affected, the OACR is loaded, and the force external function control circuit is set.

6.13.10. Load Function In Channel And Monitor

LFCM
75,11
0.75 μs – alternate
1.50 μs – same

For channel [a OR CSR]: (U) → OACR; set output active (ISI only), external function, force external function, and output monitor (ISI only).

The word from location U is transferred to the Output Access Control Register (OACR) at address 60_8 + [a OR CSR]. The output active (ISI only), external function, force external function, and output monitor (ISI only), control circuits are set for the specified channel.

(1) If D2 of PSR=1, an attempt to perform a Load Function In Channel And Monitor instruction causes a Guard Mode Fault Interrupt and none of the above steps occurs.

(2) If an ESI channel is specified for a Load Function In Channel And Monitor instruction, the output active and external function control circuits are not affected, the OACR is loaded, and the force external function control circuit is set.

(3) For an ESI channel, the Load Function In Channel And Monitor instruction performs exactly the same functions as the Load Function Channel instruction. Neither instruction affects the output monitor control circuit.

6.13.11. Jump On Function In Channel

JFC
75,12
0.75 μs

Jump to U if force external function is set for channel [a OR CSR]; go to NI if force external function is clear.

If the force external function control circuit for channel [a OR CSR] is set, U is transferred to the P register. If the force external function control circuit is in the cleared state, the next instruction (NI) is executed.

(1) If D2 of PSR = 1, an attempt to perform a Jump On Function In Channel instruction causes a Guard Mode Fault Interrupt; in this instance, the contents of the P register are not affected by the status of the force external function control circuit.

(2) This instruction tests the state of the force external function control circuit for the specified channel. This control circuit is set when a Load Function Channel or Load Function In Channel And Monitor instruction specifying that channel is performed. It is cleared during the sequence used to send the first function word for a Load Function Channel or Load Function In Channel And Monitor instruction to the peripheral subsystem attached to that channel.

6.13.12. Prevent All Channel External Interrupts

PACI
75,15
0.75 μs

Prevent all external interrupts.

This instruction sets a control circuit which prevents the I/O section from reacting to any External Interrupt or Day Clock Interrupt requests.

(1) If D2 of PSR = 1, an attempt to perform a Prevent All Channel External Interrupts instruction causes a Guard Mode Fault Interrupt and the control circuit is not set.

(2) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(3) The only effects that the values in the x, h, i, and u fields can have on the operation is the index register incrementation obtained when $x \neq 0$ and h = 1, and the indirect addressing delay introduced when i = 1 and D7 of PSR = 0. It is recommended that the x, h, i, and u fields contain all 0 bits.

(4) If interrupts are not disabled at the time the Prevent All Channel External Interrupts instruction is initiated, the I/O section will react to the highest priority Day Clock Interrupt/External Interrupt signal, should any occur during the early stages of the Prevent All Channel External Interrupts instruction, by initiating the interrupt associated with that signal immediately following the instruction. Only one such request, however, will be so honored.

(5) Once the Prevent All Channel External Interrupts instruction is performed, it is necessary to perform the Allow All Channel External Interrupts instruction to enable the CPU to react to Day Clock Interrupt/External Interrupt requests (see 6.13.13). Execution of the Allow All I/O Interrupts And Jump instruction or the Enable Day Clock instruction is not sufficient.

6.13.13. Allow All Channel External Interrupts

AACI
75,14
0.75 μs

Allow all external interrupts.

This instruction clears the control circuit set by the Prevent All Channel External Interrupts instruction (see 6.13.12) and allows the I/O section to react to Day Clock Interrupt requests and External Interrupt requests.

(1) If D2 of PSR = 1, an attempt to perform an Allow All Channel External Interrupts instruction causes a Guard Mode Fault Interrupt and the control circuit is not cleared.

(2) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(3) The only effects that the values in the x, h, i, and u fields can have on the operation is the index register incrementation obtained when $x \neq 0$ and h = 1, and the indirect addressing delay introduced when i = 1 and D7 of PSR = 0. It is recommended that the x, h, i, and u fields contain all 0 bits.

(4) The Allow All Channel External Interrupts instruction clears the circuit set by the Prevent All Channel External Interrupts instruction (see 6.13.12). It does not clear the circuit set by a Prevent All I/O Interrupts And Jump instruction (see 6.14.1) or the honoring of an interrupt request; the Allow All I/O Interrupts And Jump instruction (see 6.9.3) is required for this. The Allow All Channel External Interrupts instruction can never be used to enable the Day Clock; the Enable Day Clock instruction (see 6.14.8) is required for this.

6.14. EXECUTIVE SYSTEM CONTROL INSTRUCTIONS

The instructions in this group are intended for use solely by the Executive System. All but one (see 6.14.2) are privileged instructions; an attempt to perform a privileged instruction when D2 of PSR = 1 causes a Guard Mode Fault Interrupt.

6.14.1. Prevent All I/O Interrupts And Jump

PAIJ
72,13
0.75 μs

Prevent all I/O interrupts and jump to U.

The I/O section of the CPU will not recognize any I/O interrupt requests received following the completion of the instruction nor will it react to any I/O interrupt requests received following the start of the execution of the instruction. The instruction stored in location U is executed next.

(1) The following are classed as I/O interrupts:

- All monitor interrupts
- All external interrupts including the Day Clock Interrupt
- All system I/O interrupts including the Interprocessor Interrupt, Real Time Clock (RTC) Interrupt, and the Power Loss Interrupt
- All I/O parity error interrupts including the Access Control Word Parity Error Interrupt and the I/O Data Parity Error Interrupt.

(2) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(3) This is a privileged instruction. If D2 of PSR = 1, an attempt to perform a PAIJ instruction causes a Guard Mode Fault Interrupt.

(4) Once the Prevent All I/O Interrupts And Jump instruction is performed, an Allow All I/O Interrupts And Jump instruction (see 6.9.3) must be performed in order for the CPU to recognize and react to any I/O interrupt request. Execution of the Allow All Channel Interrupts instruction (see 6.13.13) or the Enable Day Clock instruction (see 6.14.8) is not sufficient.

(5) If an External Interrupt (EI) signal is turned on while a Prevent All I/O Interrupts And Jump instruction is being performed, the CPU may acknowledge receipt of the status word and store it immediately following completion of the Prevent All I/O Interrupt And Jump instruction. The program will not be interrupted to complete the reaction to the EI signal until after an Allow All I/O Interrupts And Jump instruction has been performed. Therefore, the contents of the status word location associated with an EI signal should not be changed in any manner following a Prevent All I/O Interrupts And Jump instruction since the location may contain a status word of which the program is unaware because the associated interrupt has not occurred.

6.14.2. Store Channel Number

SCN
72,14
0.75 μs – alternate
1.50 μs – same

If a = 0: channel number → U_{3-0};
if a = 1: channel number → U_{3-0} and CPU number → U_{5-4}.

When the a field contains all 0 bits, the 4-bit number identifying the I/O channel associated with the most recent I/O interrupt for the CPU performing the instruction is stored in bits 3 through 0 of the location U. Bits 17 through 4 of location U are cleared to 0 bits.

When the a field contains 0001_2, the 4-bit number identifying the I/O channel associated with the most recent I/O interrupt is stored in bits 3 through 0 of location U, the 2-bit number identifying the CPU is stored in bits 5 and 4 of location U, and bits 17 through 6 of location U are cleared to 0 bits.

(1) If the a field contains a value other than 0000_2 or 0001_2, the result is undefined.

(2) If U is greater than, or equal to, 200_8, the bits 35 through 18 of location U remain unchanged. If U is less than 200_8, bits 35 through 18 of the control register addressed by U are cleared to 0 bits.

(3) This is not a privileged instruction. When D2 of PSR = 1, an SCN instruction can be performed without causing a Guard Mode Fault Interrupt.

6.14.3. Load Processor State

LPS
72,15
0.75 μs – alternate
1.50 μs – same

(U) → PSR

The contents of location U are transferred to the Processor State Register (PSR).

This instruction provides the means for an Executive type program to modify the operating state of the machine. The Executive program must define a discrete Processor State Word for each user program under its control, and set the specific user mode before transferring control to that user. The Processor State Word associated with a specific program defines the various designators which control the hardware operation, and contains the proper values for the relative addressing base registers, BI, BD, and BS (see 9.1).

(1) The Load Processor State instruction which initiates the change from Executive mode to user mode is usually followed by one of the following transitional instructions:

- LPS with u and $(Xm)_X$ such that U < 200 (see note 5).
- NOP with a, x, h, and u = 0.
- J,JK with h = 0.
- AAIJ with h = 0.

When the transitional instruction is executed, the PSR values which apply (including BI, BD, BS, D7, D6, D4, D3, and D2) are the values in the PSR before the initial Load Processor State instruction. If this instruction specifies indirect addressing, the PSR values loaded by the initial Load Processor State instruction apply for all passes after the first pass.

(2) When a Load Processor State instruction is performed, the contents of the location specified by the U value are actually loaded into the PSR during cycle 3 of the T1 chain associated with the Load Processor State instruction (see 5.4.1.2). During this cycle and the cycle immediately following reading from and writing into control registers is inhibited. If the U value associated with the Load Processor State instruction references a control register, there is no Read Instruction delay between cycle 5 of the T0 chain for the initial Load Processor State instruction and the Read X_X in cycle 1 of the T0 chain for the transition instruction. This means the initial Load Processor State instruction is performed with alternate bank timing. As a result, cycle 3 of the T1 chain associated with the initial Load Processor State instruction coincides with cycle 3 of the T0 chain for the transitional instruction. Thus, the transitional instruction must be such that cycles 3 and 4 of the T0 chain do not require reading from or writing into control registers. The transitional instructions listed in note (1) comply with this restriction. Since the transitional instruction is already being processed by the time the contents of the U value for the

initial Load Processor State instruction is stored, the machine state specified by this Load Processor State instruction effectively applies to the second instruction performed following the Load Processor State instruction. This machine state then applies to all subsequent instructions until the contents of the PSR are changed by another Load Processor State instruction or by an interrupt.

(3) This is a privileged instruction. If D2 of PSR = 1, an attempt to perform a Load Processor State instruction causes a Guard Mode Fault Interrupt.

(4) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

(5) The recommended method for an executive program (D3D2 = 00, D6 = 1) to return control to a user program after handling an interrupt is to perform the following sequence of instructions with I/O interrupts locked out:

LPS with u and $(Xm)_X$ such that U < 200
LPS with h = i = 0, and u and $(Xm)_X$ such that U < 200
AAIJ with h = i = 0

The first Load Processor State instruction specifies values for u and $(Xm)_X$ such that U < 200 (that is, U addresses a control register) to ensure that the Load Processor State instruction is performed with alternate bank timing. The word transferred to the PSR specifies D7 = 0, D6 = 1, D3D2 = 00, and the user values for BI, BD, and BS. This instruction prepares for the Allow All I/O Interrupts And Jump instruction by setting up BI, BD, and BS for returning to the user program. At the same time, it retains the executive control registers so that an executive index register can be read for the Allow All I/O Interrupts And Jump instruction. The index register specified by the Allow All I/O Interrupts And Jump instruction should contain the proper user relative address. The user relative address, P + 1 or P + 2, was captured by the executive when control was transferred from that user to the executive.

The second Load Processor State instruction does not specify indirect addressing or index register incrementation. Again the values for u and $(Xm)_X$ specified by this transitional Load Processor State instruction should be such that U < 200 to ensure that this Load Processor State instruction is performed with alternate bank timing. The transitional Load Processor State instruction must be performed with alternate bank timing so that the PSR values set by the initial Load Processor State instruction will apply to the read X_X cycle of the Allow All I/O Interrupts And Jump instruction. The word transferred to the PSR specifies D7 = 0, D6 = 0, D3D2 = 01 or 11, and the same values for D1 and D0 as were captured at control register 000_8 when control was transferred from that user to the executive program. BI, BD, and BS contain the same user values as used for the first Load Processor State instruction. The second Load Processor State instruction completes the switch from the executive mode to the user mode. The user state specified by this instruction effectively applies to the instruction following the Allow All I/O Interrupts And Jump instruction.

The Allow All I/O Interrupts And Jump instruction specifies h = i = 0. When this instruction is performed, the machine state specified by the first Load Processor State instruction prevails. Thus the executive mode index register is read in the Read X_X step, but the user BI, BD, and BS are used to determine the jump to address.

(6) The Storage Limits Register must be set to the proper values for the user program before a Load Processor State instruction is executed which sets guard mode and storage limits protection for the user mode.

6.14.4. Load Storage Limits

LSL
72,16
0.75 μs – alternate
1.50 μs – same

(U) → SLR

The contents of location U are transferred to the 36-bit Storage Limits Register (SLR).

(1) Performing a Load Storage Limits instruction does not enable the storage protection capability, but merely defines or redefines the storage protection limits. If D3 of PSR = 1, write-only storage protection is provided for the operand for the next instruction. If D3 of PSR = 0, the storage protection feature is not enabled until after the PSR is subsequently loaded with a value which provides a 1 bit for either D2 or D3 or both. For more details on the Storage Limits Register see 9.3.1.

(2) This is a privileged instruction. If D2 of PSR = 1, an attempt to perform a Load Storage Limits instruction causes a Guard Mode Fault Interrupt.

(3) The contents of the a field are ignored; however, it is recommended that the a field contain all 0 bits.

6.14.5. Initiate Interprocessor Interrupt

III
73,14 a = 0 or 1
0.75 μs

Initiate interprocessor interrupt.

The value in the a field is used to determine which processor in a multiprocessor system is to be interrupted, and the Interprocessor Interrupt signal is turned on for the specified processor.

(1) In a multiprocessor system, the CPU connected to the active Interprocessor Interrupt signal is interrupted. The active interrupt signal can be determined by forming the base 3 sum of the contents of a + 1 + CPU performing the Initiate Interprocessor Interrupt instruction. The interrupt location is determined from the a field of the Initiate Interprocessor Interrupt instruction. If $a = 0000_2$, the interrupt is to location 233_8 + MSR. If $a = 0001_2$, the interrupt is to location 232_8 + MSR. This is summarized in Table 6-2.

CPU # PERFORMING III INSTRUCTION	CONTENTS OF A FIELD OF III INSTRUCTION	INTERRUPTED CPU #	INTERRUPT LOCATION (OCTAL)
0	0000 0001*	1 2	233 + MSR 232 + MSR
1	0000* 0001	2 0	233 + MSR 232 + MSR
2	0000 0001	0 1	233 + MSR 232 + MSR

* If the system includes only two CPU's, the results are not defined if an Initiate Interprocessor Interrupt instruction is performed by CPU #0 with a = 0001 or by CPU #1 with a = 0000.

Table 6-2. Interprocessor Interrupt Relationships

(2) This is a privileged instruction. If D2 of PSR = 1, an attempt to perform an Initiate Interprocessor Interrupt instruction causes a Guard Mode Fault Interrupt.

(3) The only effects that the contents of the x, h, i, and u fields can have on the operation are the index register incrementation obtained when $x \neq 0$ and $h = 1$, and the indirect addressing delay introduced when $i = 1$ and D7 of PSR = 0. It is recommended that the x, h, i, and u fields contain all 0 bits.

(4) When $f = 73_8$ and $j = 14_8$, the value in the a field is used to specify the instruction, as follows:

(a) When $a = 1000_2$, the Alarm instruction (see 6.14.6) is specified.

(b) When $a = 0000_2$ or 0001_2, the Initiate Interprocessor Interrupt instruction is specified.

(c) When $a = 1010_2$, the Disable Day Clock instruction (see 6.14.7) is specified.

(d) When $a = 1001_2$, the Enable Day Clock instruction (see 6.14.8) is specified.

(e) The results are undefined if the a field contains a value in the range 0010_2 through 0111_2 or 1011_2 through 1111_2.

6.14.6. Alarm

ALRM
73,14 $a = 10_8$
0.75 μs

Turn on alarm.

The audio alarm on the operator's Display Console is turned on.

(1) The audio alarm may be turned off by pressing the ALARM RESET switch on the operator's Display Console or on the maintenance panel.

(2) This is a privileged instruction. If D2 of PSR = 1, an attempt to perform an alarm instruction causes a Guard Mode Fault Interrupt.

(3) The only effects that the values in the x, h, i, and u fields can have on the operation is the index register incrementation obtained when $x \neq 0$ and $h = 1$, and the indirect addressing delay introduced when $i = 1$ and D7 of PSR = 0. It is recommended that the x, h, i, and u fields contain all 0 bits.

(4) When $f = 73_8$ and $j = 14_8$, the value in the a field is used to specify the instruction, as follows:

(a) When $a = 1000_2$, the Alarm instruction is specified.

(b) When $a = 0000_2$ or 0001_2, the Initiate Interprocessor Interrupt instruction (see 6.14.5) is specified.

(c) When $a = 1010_2$, the Disable Day Clock instruction (see 6.14.7) is specified.

(d) When $a = 1001_2$, the Enable Day Clock instruction (see 6.14.8) is specified.

(e) The results are undefined if the a field contains a value in the range 0010_2 through 0111_2 or 1011_2 through 1111_2.

6.14.7. Disable Day Clock

DDC
73,14 $a = 12_8$
0.75 μs

Disable day clock.

A control circuit is set within the CPU to cause the CPU's I/O section to ignore request and interrupt signals from the Day Clock on the operator's Display Console associated with the CPU executing the instruction.

(1) After the Disable Day Clock instruction has been performed, an Enable Day Clock instruction (see 6.14.8) must be performed in order for the CPU to react to either a Day Clock Request or a Day Clock Interrupt signal. Performing an Allow All I/O Interrupts And Jump (see 6.9.3) or an Allow All Channel External Interrupts instruction (see 6.13.13) is not sufficient.

(2) This is a privileged instruction. If D2 of PSR = 1, an attempt to perform a Disable Day Clock instruction causes a Guard Mode Fault Interrupt.

(3) The only effects that the values in the x, h, i, and μ fields can have on the operation is the index register incrementation obtained when $x \neq 0$ and $h = 1$, and the indirect addressing delay introduced when $i = 1$ and D7 of PSR = 0. It is recommended that the x, h, i, and u fields contain all 0 bits.

(4) When $f = 73_8$ and $j = 14_8$, the value in the a field is used to specify the instruction, as follows:

(a) When $a = 1000_2$, the Alarm instruction (see 6.14.6) is specified.

(b) When $a = 0000_2$ or 0001_2, the Initiate Interprocessor Interrupt instruction (see 6.14.5) is specified.

(c) When $a = 1010_2$, the Disable Day Clock instruction is specified.

(d) When $a = 1001_2$, the Enable Day Clock instruction (see 6.14.8) is specified.

(e) The results are undefined if the a field contains a value in the range 0010_2 through 0111_2 or 1011_2 through 1111_2.

6.14.8. Enable Day Clock

EDC
73,14 $a = 11_8$
0.75 μs

Enable day clock.

The control circuit which is set by a Disable Day Clock instruction is cleared. This permits the CPU's I/O section to recognize and react to Day Clock Request signals and enables a circuit which causes the CPU's I/O section to respond to Day Clock Interrupt signals.

(1) The Enable Day Clock instruction enables the circuit disabled by the Disable Day Clock instruction (see 6.14.7). It does not enable the circuit disabled by a Prevent All I/O Interrupts And Jump instruction (see 6.14.1) or the honoring of an interrupt request; the Allow All I/O Interrupts And Jump instruction (see 6.9.3) is required for this. It does not enable the circuit disabled by the Prevent All Channel External Interrupts instruction (see 6.13.12); the Allow All Channel External Interrupts instruction (see 6.13.13) is required for this.

(2) This is a privileged instruction. If D2 of PSR = 1, an attempt to perform an Enable Day Clock instruction causes a Guard Mode Fault Interrupt.

(3) The only effects that the values in the x, h, i, and u fields can have on the operation is the index register incrementation obtained when $x \neq 0$ and $h = 1$, and the indirect addressing delay introduced when $i = 1$ and D7 of PSR = 0. It is recommended that the x, h, i, and u fields contain all 0 bits.

(4) When $f = 73_8$ and $j = 14_8$, the value in the a field is used to specify the instruction, as follows:

(a) When $a = 1000_2$, the Alarm instruction (see 6.14.6) is specified.

(b) When $a = 0000_2$ or 0001_2, the Initiate Interprocessor Interrupt instruction (see 6.14.5) is specified.

(c) When $a = 1010_2$, the Disable Day Clock instruction (see 6.14.7) is specified.

(d) When $a = 1001_2$, the Enable Day Clock instruction is specified.

(e) The results are undefined if the a field contains a value in the range 0010_2 through 0111_2 or 1011_2 through 1111_2.

6.14.9. Select Interrupt Locations

SIL
73,15
0.75 μs

(a) → MSR

The three low order bits of the a field are transferred to the Memory Select Register (MSR). MSR OR MSR SWITCH SETTINGS on the Maintenance Panel bias all fixed address assignment references.

(1) The new value loaded in the MSR by the Select Interrupt Locations instruction is immediately available in the MSR to determine the absolute address of all fixed address assignment references.

(2) Permissible MSR values are given in 3.2.3.2. The value chosen must correspond to a main storage module or module pair which is physically included in the system.

(3) This is a privileged instruction. If D2 of PSR = 1, an attempt to perform a Select Interrupt Locations instruction causes a Guard Mode Fault Interrupt.

(4) The leftmost bit of the a field is ignored by the control section. The only effects that the contents of the x, h, i, and u fields can have on the operation are the index register incrementation obtained when $x \neq 0$ and $h = 1$, and the indirect addressing delay introduced when $i = 1$ and D7 of PSR = 0. It is recommended that the leftmost bit of the a field and the x, h, i, and u fields contain all 0 bits.

6.14.10. Load Channel Select Register

LCR
73,16 a = 0
0.875 μs – alternate
1.625 μs – same

$(U)_{3-0}$ → CSR

The contents of bits 3 through 0 of location U are transferred to the 4-bit Channel Select Register (CSR).

(1) The new value loaded in the CSR by the Load Channel Select Register instruction is available for use by the next instruction. Additional information on the CSR is given in 6.13.

(2) The CSR is cleared to contain all 0 bits when the CPU is manually master cleared.

(3) This is a privileged instruction. If D2 of PSR = 1, an attempt to perform a Load Channel Select Register instruction causes a Guard Mode Fault Interrupt.

(4) When $f = 73_8$ and $j = 16_8$, the Load Channel Select Register instruction is specified when the a field contains 0000_2. The Load Last Address Register instruction (see 6.14.11) is specified when the a field contains 0001_2.

The results are undefined if the a field contains a value in the range 0010_2 through 1111_2.

6.14.11. Load Last Address Register

LLA
73,16 a = 1
0.875 μs – alternate
1.625 μs – same

$(U)_{2-0} \rightarrow$ LAR

The contents of bits 2 through 0 of location U are transferred to the 3-bit Last Address Register (LAR).

(1) The new value loaded in the LAR by the Load Last Address Register instruction is available for use in the event that a main storage parity error is detected during the execution of the next instruction (or a subsequent instruction). If a main storage parity error (see 8.3.1.1) is detected in logical module pair #0 (for noninterleaved main storage) or in logical module #0 (for noninterleaved main storage), bits 17 through 15 of the absolute address of the interrupt location are determined by forming the logical OR of the contents of the LAR and the leftmost three Last Address switches on the CPU's maintenance panel. (See Table 8–2.)

(2) The LAR is cleared to contain all 0 bits when the CPU is manually master cleared.

(3) This is a privileged instruction. If D2 of PSR = 1, an attempt to perform a Load Last Address Register instruction causes a Guard Mode Fault Interrupt.

(4) When $f = 73_8$ and $j = 16_8$, the Load Last Address Register instruction is specified when the a field contains 0001_2. The Load Channel Select Register instruction (see 6.14.10) is specified when the a field contains 0000_2. The results are undefined if the a field contains a value in the range 0010_2 through 1111_2.

6.15. ILLEGAL FUNCTION CODES

An attempt to perform an instruction containing any one of the following function codes or f, j combinations causes an Illegal Instruction Fault Interrupt:

00

07

33

37

72,00

72,12

72,17

77,00–17

An attempt to perform a Test and Set instruction by a CPU of a nonexpendable UNIVAC 1108 Unit Processor System (Type 3011-99; see 1.3.1) also causes an Illegal Instruction Fault Interrupt.

When an Illegal Instruction Fault Interrupt occurs, the instruction in location 241_8 + MSR is performed next (see 8.3.2.1). Except for the case of f = 77 and f,j = 73,17 if the CPU is in the 1108 Mode (D4 of PSR = 0) and not in guard mode (D2 of PSR = 0) when an Illegal Instruction Fault Interrupt occurs, the System Fault Alarm is turned on. This alarm is turned off by depressing the ALARM RESET switch on the operator's console or on the maintenance panel.

An attempt to perform an instruction containing any one of the following, f,j combinations when the CPU is in guard mode (D2 of PSR = 1) causes a Guard Mode Fault Interrupt:

75,13

75,16

75,17

When a Guard Mode Fault Interrupt occurs, the instruction in location 243_8 + MSR is executed next.

7. INPUT/OUTPUT

7.1. INTRODUCTION

The input/output (I/O) section of the UNIVAC Processor Unit (CPU) controls the transfer of data between main storage and a subsystem over the input and output channels. The I/O section can control a maximum of 16 I/O channels.

When an input operation is performed, either input or status words are transferred from a subsystem to main storage. When an output operation is performed, either function or output words are transferred from main storage to a subsystem. Bi-directional transfers never occur simultaneously. The word length of all transferred words is 36 bits.

7.1.1. I/O Channel Interface

Figure 7–1 shows the control and data signal lines for an I/O channel. Table 7–1 lists the function of each type of line.

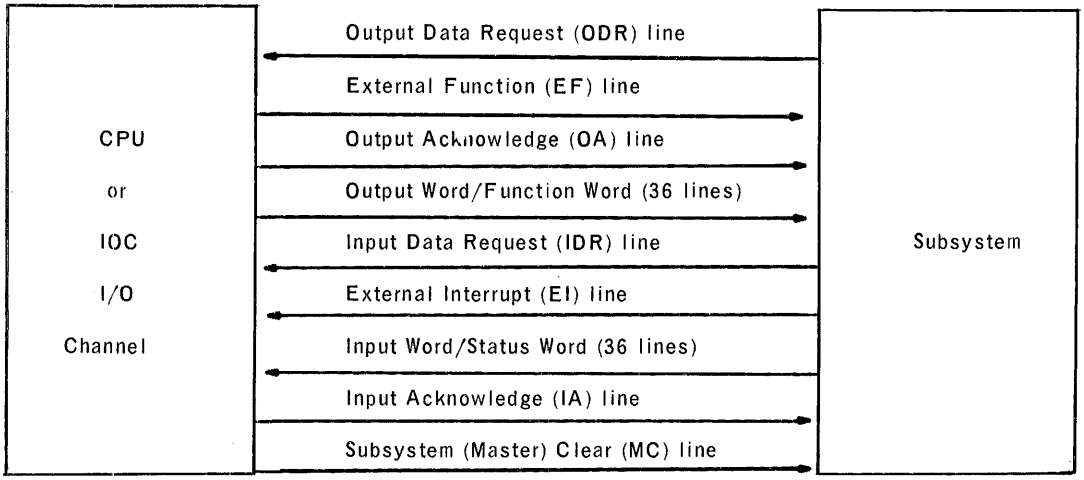


Figure 7–1. I/O Channel Interface, Control and Data Lines

TYPE OF LINE	NUMBER OF LINES	FUNCTION
Output Data Request (ODR)	1	Transmits a control signal from the subsystem which indicates that the subsystem is ready to receive an output word or a function word. When the subsystem turns on the ODR, it must hold it on until after an output word or function word has been received. Exception: the first word transferred to a subsystem after activation of the channel in function mode. See 7.2.10.
External Function (EF)	1	Transmits a control signal to the subsystem which indicates to the subsystem that signals for a function word (or search value word) are on the output word/function word lines. The EF control signal is a single pulse. 200 ns
Output Acknowledge (OA)	1	Transmits a control signal to the subsystem which indicates to the subsystem that signals for an output word are on the output word/function word lines. The OA control signal is a single pulse. 200 ns
Output Word Function Word	36	Transmits word signals to the subsystem. These signals represent output words or function words (or search value words) of up to 36 bits in length each. no latch. depend on data only during EF or OA
Input Data Request (IDR)	1	Transmits a control signal from the subsystem which indicates that the subsystem is presenting an input word on the input word/status word lines. When the subsystem turns on the IDR control signal it must hold it on until the input acknowledge signal is received from the CPU. See 7.2.10.
External Interrupt (EI)	1	Transmits a control signal from the subsystem which indicates that the subsystem is presenting a status word on the input word/status word lines. When the subsystem turns on the EI control signal it must hold it on until the input acknowledge signal is received from the CPU. See 7.2.10.
Input Word/Status Word	36	Transmits word signals from the subsystem. These word signals represent input words or status words of up to 36 bits in length each. The subsystem must hold the word signals on the input word/status word lines until the input acknowledge signal is received from the CPU. must be maintained during IA.
Input Acknowledge (IA)	1	Transmits a control signal from the CPU to the subsystem which indicates to the subsystem that input word or status word from the subsystem has been accepted. The IA control signal is a single pulse. 200 ns
Subsystem MASTER Clear (MC)	1	Transmits a control signal to the subsystem. The subsystem responds to this control signal by stopping all subsystem activity and setting the ODR signal on. The MC control signal is turned on by manually depressing either the I/O CLEAR switch located on the CPU's maintenance panel or the SUB - SYSTEM CLEAR switch located on the Display Console. The MC control signal can also be turned on through operation of the Availability Control Unit (ACU). 100 ms

Table 7–1. I/O Channel, Control and Data Lines

7.1.2. I/O Channel Numbering and Configurations

A CPU can have either 8, 12, or 16 I/O channels. If a CPU has 8 I/O channels, the channel numbers assigned are 0 through 3 and 12 through 15. If a CPU has 12 I/O channels, the channel numbers assigned are either 0 through 3 and 8 through 15 or 0 through 7 and 12 through 15. If a CPU has 16 I/O channels, the channel numbers assigned are 0 through 15. I/O channel 15 is always used for the Display Console and the day clock.

7.1.3. ISI Versus ESI Mode of I/O Channel Operation

I/O channels 0 through 14 may be assigned to operate either in the Internally Specified Index (ISI) mode or the Externally Specified Index (ESI) mode. ISI/ESI mode selection is made by 15 2-position switches located on the CPU maintenance panel. Each of the 15 switches is assigned to one of the I/O channels (0 through 14); no ISI/ESI switch is provided for I/O channel 15.

The details of the ISI mode of I/O channel operation are explained in 7.2; the ESI mode of I/O channel operation is explained in 7.3. In ISI mode only one subsystem is interfaced to a channel, but many peripheral devices may be interfaced through a suitable multiplexing device on a channel operating in the ESI mode.

7.1.4. ESI Mode – Half Word Versus Quarter Word Operation

I/O channels in ESI mode can operate with input or output word interfaces which are either half word or quarter word. When the half word interface is used, 18-bit input or output words are transmitted from or to the subsystem via that I/O channel. When the quarter word interface is used, 9-bit input or output words are transmitted from or to the subsystem via that I/O channel. A patch card, used to select either half word or quarter word operation, is supplied for each I/O channel.

For half word ESI operation, two I/O words (18 bits each) are stored in each main storage location associated with that ESI mode I/O word transfer operation. For quarter word ESI operation, four I/O words (9 bits each) are stored in each main storage location associated with that ESI mode I/O word transfer operation.

7.1.5. Normal/Compatible I/O Channels

The UNIVAC 1108 System is capable of operating with the UNIVAC 1107 and the UNIVAC 1108 Subsystems by using the normal/compatible channel option (needed to interface with some UNIVAC 1107 Subsystems).

Two switches, located on the CPU maintenance panel, are used to select either normal or compatible operation of the CPU's I/O channels. One switch controls the selection for I/O channels 0 through 7; the other switch controls the selection for I/O channels 8 through 15.

The maximum instantaneous word transfer rate that can be achieved on one normal channel under ideal conditions is 444,000 words per second (one word per 2.25 microseconds). The maximum instantaneous word transfer rate which can be achieved on a system basis on more than one normal channel under ideal conditions is 1,333,333 words per second (one word per 750 nanoseconds).

The length of time the I/O section holds control and word signals on the interface lines of a normal channel is shorter than that required for some of the subsystems used with the UNIVAC 1107 CPU. Therefore, to use these UNIVAC 1107 Subsystems with the UNIVAC 1108 the compatible channel operation is provided, which lengthens the time the control and word signals are held on channel interface lines. The maximum instantaneous word transfer rate that can be achieved on a compatible channel or a group of compatible channels is 200,000 words per second (one word per 5 microseconds).

NOTE: The Display Console and day clock on channel 15 operates equally well regardless of whether the channel is in the normal or compatible mode.

7.1.6. I/O Channel Activity – Introduction

A channel is activated upon execution of an appropriate instruction by the CPU's main control section. The input channel is activated by the Load Input Channel (LIC) and Load Input Channel And Monitor (LICM) instructions. The output channel is activated by the Load Output Channel (LOC) and Load Output Channel And Monitor (LOCM) instructions. The Load Function Channel (LFC) and Load Function Channel And Monitor (LFCM) instructions activate an output channel only if the specified output channel is operating in ISI mode; these instructions do not activate the specified channel if that channel is in ESI mode.

In ISI mode, each channel activating instruction specifies an operand which is an Access Control Word (ACW). The ACW is loaded into the control register associated with the specified channel for input or output. The ACW is then referenced by the CPU to determine the main storage location of each word to be transferred, and for a count of the total number of words to be transferred during this operation.

In the absence of I/O activity, the CPU continuously executes programs. While executing instructions, the CPU also checks for required I/O activity each 125 microseconds. The sequence of operations for a single data word input transfer is as follows:

During one of the cyclic checks for I/O activity, the CPU recognizes that input is active for the specified channel, the IDR signal from this channel's subsystem is on, and this input data transfer is the highest priority I/O operation requiring service. The coincidence of these signals forces the CPU to momentarily stop instruction processing while the input data transfer occurs. The CPU reads the contents of the Input Access Control Register (IACR) for this channel. It samples the input data lines for the channel and places the data word in an internal register. The ACW provides the address in main storage for the input data word, and a main storage request is initiated to write the data word from the internal register into the specified main storage location. An Input Acknowledge signal is sent to the subsystem which informs the subsystem that the input data word has been received, and allows the subsystem to continue processing.

The ACW main storage address is incremented by one, the word count portion of the ACW is decremented by one, and the updated ACW is stored back into the IACR for this channel. The CPU then reverts to instruction processing, or should there be additional I/O activity, continues with similar I/O sequences.

Fields within the ACW provide additional controls for directing placement of words in storage (see 7.2 and 7.3). The input activity for this channel terminates when the CPU recognizes that all words specified in the word count field of the ACW have been input.

A typical output data sequence is initiated when the CPU recognizes the coincidence of an output channel active, the ODR signal is on for this channel, and the priority of I/O activities is appropriate for an output data transfer. At the beginning of the output data sequence, the CPU reads the Output Access Control Word (OACW) from the appropriate control register. The address field of the ACW specifies the main storage location from which the data word is retrieved. The data word is presented to the sybsystem on the output data lines for the channel. An OA signal is sent to the subsystem indicating that the data is present on the output lines and should be sampled by the subsystem. The ACW is updated by decrementing the word count and incrementing the address. The updated ACW is stored in the control register.

The subsystem will turn off its ODR signal while it processes the newly received word. When the subsystem again requires an output word, it turns on the ODR. A channel in ISI mode sends function words in the same manner using the same OACR as that employed for output data. Each Function Word output is accompanied by the EF signal.

The first function word transmission to a subsystem upon execution of the LFC or LFCM instruction is forced. The CPU's I/O section simulates an ODR and the first function word is transmitted to the subsystem regardless of whether or not the subsystem has turned on an ODR. This forced function word is accompanied by the EF signal. If more than one function word is to be transmitted to the subsystem, the additional words are controlled by the ACW.

When an output channel is in ESI mode, execution of the LFC or LFCM instruction does not activate that channel or the monitor function on that channel. (The operations of the LFC and LFCM instructions are equivalent.) However, one function word is transmitted from main storage to a peripheral device via that channel each time the LFC or LFCM instruction is executed.

The ACW format for ISI operations is presented in 7.2.3. In the ESI mode of operation, many peripheral devices can be interfaced to the CPU through each input/output channel and separate ACW's located in main storage are associated with the input/output word transfer operations for each peripheral device. For ESI operations, two different formats may be used: one for half word ESI operations and one for quarter word ESI operations. The half word ESI ACW format is presented in 7.3.3.1; the quarter word ESI ACW format is presented in 7.3.3.2.

7.1.7. Input/Output Priority Control

The I/O section of the 1108 CPU sequentially processes the I/O requests of the active subsystems or peripheral devices attached to its channels. A minimum of 750 nanoseconds is required to transfer each word to or from main storage. Service requests for the CPU channels are recognized by the coincidence of IDR signals and the input active state, ODR signals and the output active state, or EI signal.

A priority network exists to select the service request to be performed. This selection must consider the type of function to be performed, timing considerations implicit in the operation, and selection of a channel from all those requesting service.

Input operations on normal channels can proceed at the minimum cycle time of 750 nanoseconds. The duration of output signals exceeds this minimum time and certain elements of the output section must remain active for a longer period. To lessen the delay in servicing successive output requests, the CPU channels are divided into two subgroups (channels 0 through 7 and 8 through 15). Only one output operation can occur on a subgroup in each 1.5 microseconds. Output on the two subgroups, however, can be initiated in two adjoining 750 nanosecond periods.

Compatible channel signals require more time than that needed for normal channel operation. Successive service requests for compatible channel operations are spaced by inserting at least one 750 nanosecond instruction cycle between each compatible channel operation. Operation of the I/O section of the CPU stops CPU main storage access requests. Main storage in a unit processor configuration is therefore always available to service the input or output operation. In a multiprocessor configuration, main storage access requests by other processors may conflict with the I/O section access and additional delays encountered.

Word transfers for input or output are granted higher priority in all cases than the status word input transfer associated with an EI. This priority relationship is explained in 8.2.5.

Equality of access for all active subsystems, and the maximum utilization of the input/output section requires an interaction in the priority of input and output operations. Under conditions of constant load, priority is alternately given to input and then output. In one cycle, an output is initiated. During the following 750 nanosecond cycle priority is given to an input operation. This relationship is further modified by the two channel groups which do allow simultaneous output in succeeding cycles, one in each.

The final step in selection is finding the lowest numbered channel requesting the selected function. Of all channels requesting a particular service, priority is always given to the channel having the lowest number. The priority network is defined in the following paragraphs for the two forms of channel and the interaction of the various input/output constraints.

The following rules specify the priority for operations on all active channels in the normal mode as determined by the normal compatible switches of the CPU's I/O section.

(1) An output or function word transfer has priority over all other transfers, unless an output transfer was initiated during the previous cycle.

(2) If an output transfer was initiated during the preceding cycle, priority is given to an input transfer on channels 0 through 3.

(3) If the prior cycle was not used to initiate an output operation for channels 0 through 7, and an input transfer on channels 0 through 3 is not required, then an output transfer will be initiated on the lowest numbered channel 0 through 7 requesting output service.

(4) If the prior cycle did not initiate an output transfer on channels 8 through 15, and an output on channel 0 through 7 or an input on channels 0 through 3 are not required, then an output transfer will be initiated for the lowest numbered channel 8 through 15 requesting output service.

(5) In the absence of output service requests, an input transfer will be initiated on the lowest numbered channel 0 through 15 requesting input service.

Figure 7-2 summarizes the priority for word transfer operations when all active channels are operating in the normal mode.

The following rules specify priority for word transfer operations when all active channels are operating in the compatible mode as determined by the normal/compatible switches of the CPU's I/O section.

(1) When simultaneous requests for input and output exist, priority alternates between the two operations. A designator set during each operation forces priority to be given to the alternate operation in the succeeding I/O cycle.

(2) Among all outstanding service requests, the word transfer in the selected direction is accomplished on the lowest numbered channel requesting service.

(3) Successive I/O cycles cannot be used to service compatible channels, and at least one T0 timing chain cycle in CPU main control must occur before another compatible channel service request is honored.

Figure 7-3 summarizes the priority for word transfer operations when all active channels are operating in the compatible mode.

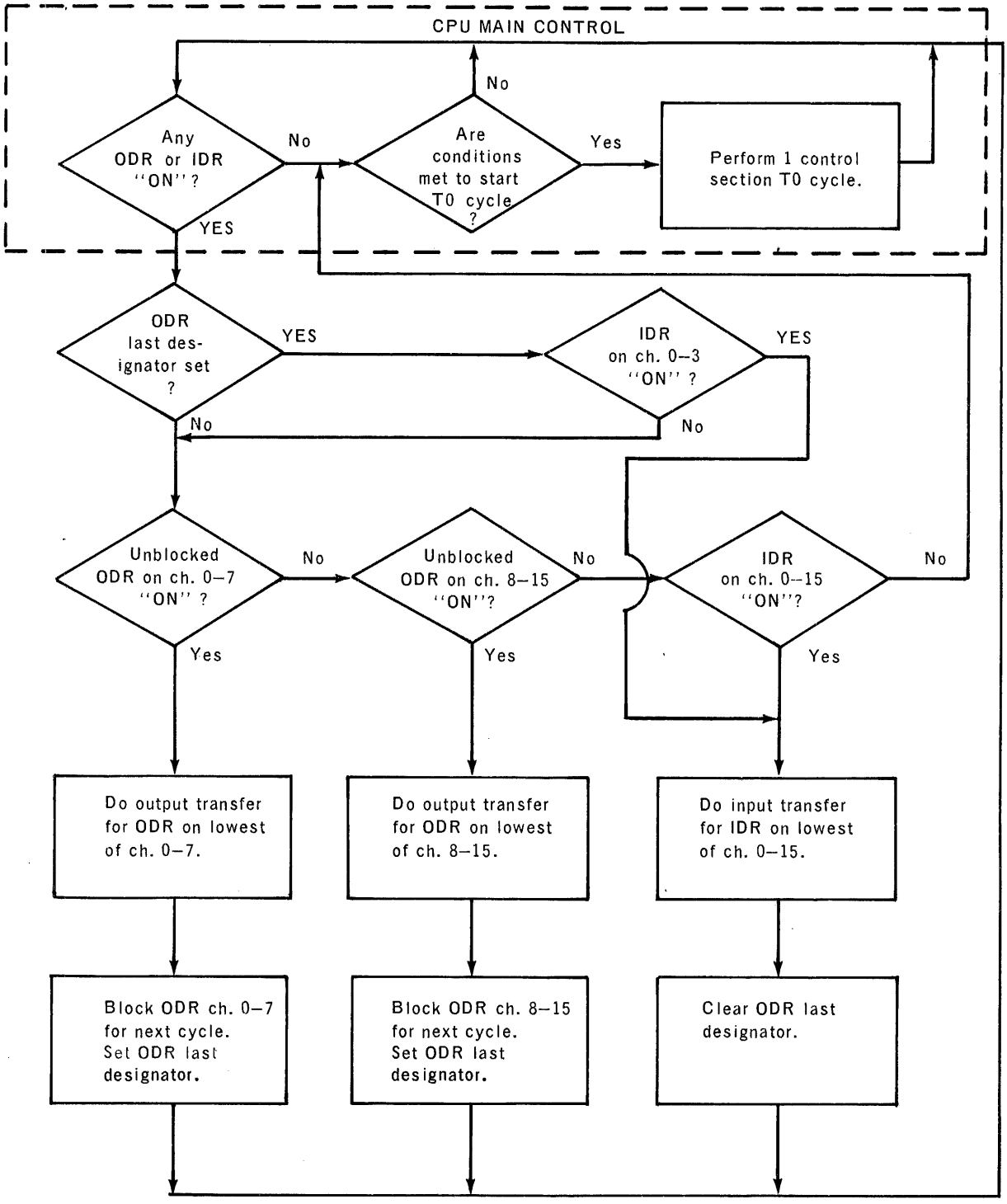


Figure 7–2. I/O Word Transfer Priority – Normal Channels Only

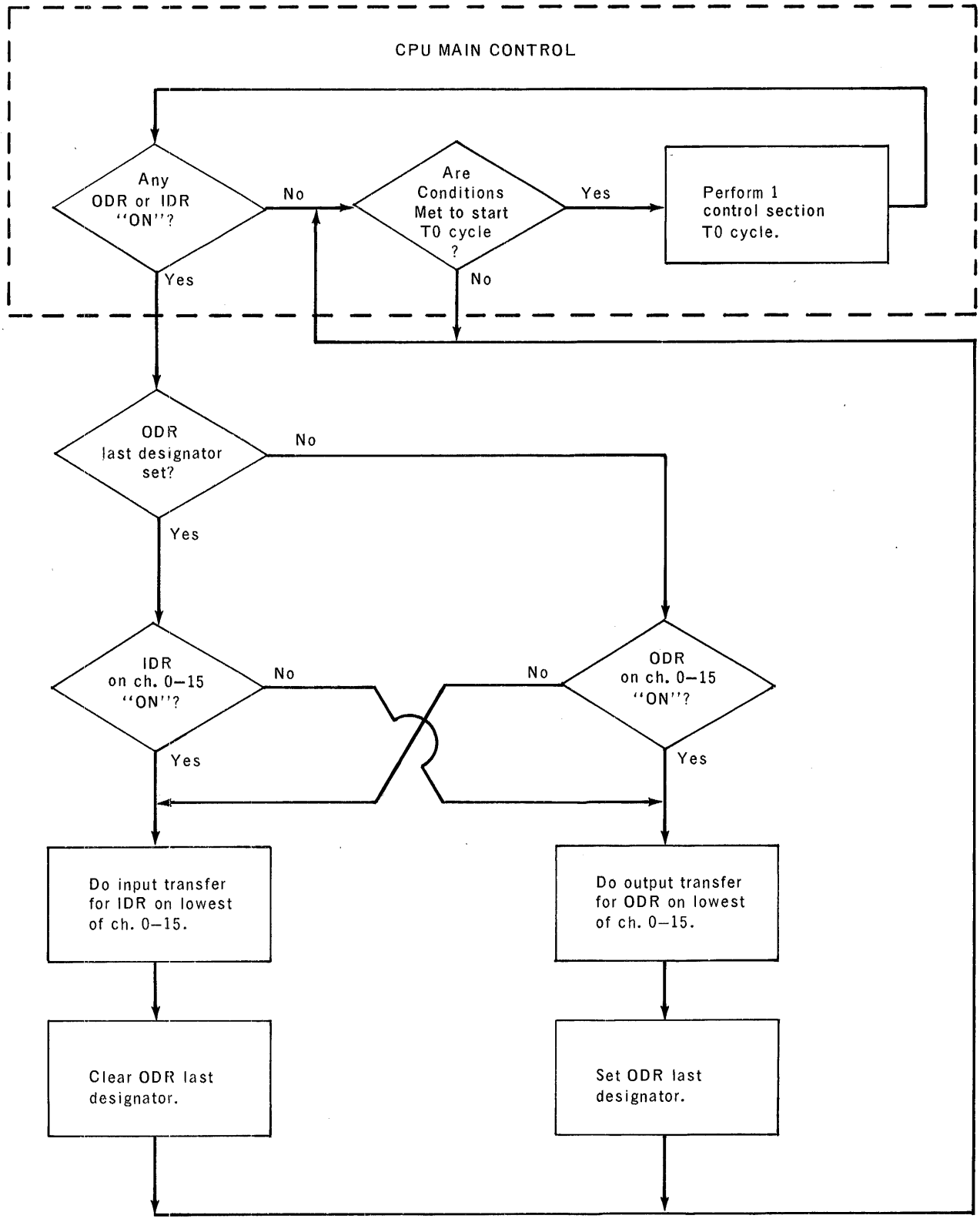


Figure 7–3. I/O Word Transfer Priority – Compatible Channels Only

7.1.8. I/O Section and Main Control Section Interaction

The I/O and main control sections of the CPU share the single main storage access logic. Input/output operations have priority over instruction processing whenever service requests for channels can be processed. The T0 timing chain for main control and the T8 timing chain for I/O are the primary control chains which initiate either form of operation. Only one of these two chains can be active at any time.

The start of the T0 timing chain in main control begins processing of a new instruction. T0 controls control register access, instruction decoding, initiation of main storage data accesses, and initiates the next instruction fetch if overlapped operation is possible. Not all parts of an instruction execution are complete at the end of the T0 chain. Other chains extending in time beyond the T0 chain provide control of extended arithmetic processing and the storing of results into control registers. Except for extended duration instructions the next T0 chain sequence can begin immediately following the prior T0 chain. In the absence of I/O, this sequencing would continue with each T0 chain marking the beginning of a new instruction processing cycle.

The T8 I/O timing chain can be started at the same points in time at which the T0 chain is started, if an I/O service request and priority network selection has occurred prior to this time. If an ODR, IDR, or EI is recognized at least 250 nanoseconds prior to the point at which the T8 chain can be started, then instruction processing will be momentarily suspended in favor of this I/O request chain. Repeated T8 cycles will occur until all I/O service requests have been satisfied. If the I/O priority network is again requesting main storage access at least 250 nanoseconds prior to the end of the current T8 I/O timing chain cycle, the I/O timing chain is activated for another service cycle. Compatible I/O channel service requests cannot be handled at this rate and these T8 timing chain cycles are usually followed by at lease one cycle of the T0 timing chain.

7.1.9. I/O Section Versus Main Control Section Main Storage Access

Both the I/O section and the main control section of the CPU can independently request access to main storage. The I/O section requests access to:

(1) transfer input words or status words into main storage, or

(2) transfer output words or function words from main storage. The main control section requests access to:

    (a) transfer instruction words from main storage to the control section, or

    (b) transfer operand words into main storage, or

    (c) transfer operand words from main storage.

The sequence of events performed when I/O words (input, status, output or function words) are transferred to or from main storage is controlled by the I/O timing chain. The main control section timing chain (T0) (see 5.4) controls the sequence of events performed when an instruction word is transferred from main storage to the main storage to the main control section and the sequence of events performed in the execution of each instruction word. The main control timing chain includes the control of operand transfers to or from control registers and main storage but does not include the events performed in the arithmetic sequence, which is controlled by the arithmetic timing chain (T4), and the storing of arithmetic results in the control registers which is controlled by the T1 timing chain. The I/O timing chain and the main control timing chain T0 are never simultaneously active. However, the T0 or I/O timing chain can be active simultaneously with the T1 timing chain or the arithmetic timing chain (T4).

The I/O timing chain has higher priority than the T0 timing chain in that the T0 timing chain is not reactivated at the end of its current cycle but the I/O timing chain is activited at that point in time if the I/O section's priority network requests main storage access at least 250 nanoseconds (two CPU main control clock cycles) prior to the end of the currently active T0 timing chain cycle. If at least 250 nanoseconds prior to the end of the current I/O timing chain cycle the I/O section's priority network is again requesting main storage access (usually for a normal I/O channel operation), the I/O timing chain is activated for another cycle upon completion of its current cycle. Each I/O timing chain cycle for a compatible I/O channel operation is usually followed by at least one cycle of the T0 timing chain.

The execution time of many arithmetic instructions is extended through the necessary activity of the arithmetic timing chain (T4). The I/O timing chain can be simultaneously active with the T4 timing chain and when the I/O timing chain is not active, the T0 timing chain can be simultaneously active with the T4 timing chain. (In this case, the T0 timing chain activity is controlling the transfer of the next instruction from main storage to the control section.)

If the I/O section's priority network does not request main storage access at least 250 nanoseconds before the end of the current T0, I/O, or T4 timing chain cycle, then upon completion of the current timing chain cycle the T0 timing chain is activated.

NOTE: If neither the T0 nor the I/O timing chain is active during the extended activity of the T4 timing chain, then the I/O timing chain is activated 250 nanoseconds after the I/O section's priority network requests main storage access provided the request occurred at least 250 nanoseconds prior to the end of the T4 timing chain cycle. Since the time necessary to complete the T4 cycle for some instructions is very long, during the execution of these instructions many cycles of the I/O timing chain can be performed during the T4 timing chain cycle.

The I/O timing chain cycle can be completed in a minimum of 750 nanoseconds. This minimum time is equal to the basic main storage cycle time. In a multiprocessor system, however, when an I/O word is transferred to or from main storage, the I/O timing chain cycle can be extended beyond the minimum of 750 nanoseconds in increments of 125 nanoseconds. The number of 125 nanosecond increments in the time extension depends on the amount of time it is necessary to expend waiting to gain access to a main storage location through a Multi-Module Access Unit (MMA) in order to transfer an I/O word to or from that addressed location.

7.2. ISI MODE - I/O OPERATION

The ISI mode of operation is employed for standard peripheral subsystems. Some of the standard peripheral subsystems are: magnetic tape, magnetic drum, high speed printer, punched card, etc.

7.2.1. General Description – Programmed Activation of an I/O Channel in ISI Mode

A channel operating in ISI mode is activated when any one of the following instructions is performed:

Load Input Channel (LIC)

Load Input Channel and Monitor (LICM)

Load Output Channel (LOC)

Load Output Channel and Monitor (LOCM)

Load Function in Channel (LFC)

Load Function in Channel and Monitor (LFCM)

The particular channel activated is determined from the logical sum of the contents of the a field specified by the instruction and the contents of the Channel Select Register (a OR CSR).

When an I/O channel is operating in ISI mode and the LIC or LICM instruction is executed, an ACW at storage location U is transferred to the IACR associated with the input channel specified by the instruction. The specified channel is then activated in input mode. When the LICM instruction is executed, the monitor function is also activated for that input channel. Upon execution of the LOC or LOCM/LFC or LFCM instruction, an ACW at storage location U is transferred to an OACR associated with the output channel specified by the instruction and that specified channel is then activated in output mode/function mode. When the LOCM/LFCM instruction is executed, the monitor function is also activated for that output channel. The ACW's that are transferred to the IACR's and OACR's are more specifically called Input Access Control Words (IACW's) and Output Access Control Words (OACW's) respectively.

While a channel is active and operating in ISI mode, full words (36 bits each – input, output or function words) are always transmitted (bit parallel, word serial) between main storage and a subsystem. The transmission of these words is controlled by the CPU's I/O section which responds to IDR's and ODR's turned on by the subsystem, or in response to an ODR simulated within the CPU's I/O section (one ODR is simulated following each execution of the LFC or LFCM instruction).

Throughout the duration of activity of an I/O channel operating in ISI mode, the address of each location in main storage to or from which a word is transferred is specified by the current contents of the IACR and OACR associated with the active channel. As each input or output function word is transferred to or from a location in main storage, the contents of the associated IACR or OACR is modified and is tested for the terminal condition (see 7.2.4). When the terminal condition is detected in the contents of an IACR and OACR, the associated active channel is automatically deactivated and the word transmissions on that channel are discontinued. If the monitor function is active on a channel, then, at the same time that the channel is deactivated upon detection of the terminal condition, the Input Monitor Interrupt, Output Monitor Interrupt or Function Monitor Interrupt occurs respectively.

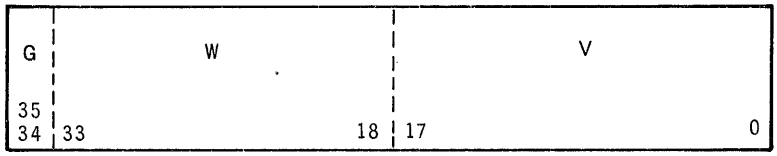
7.2.2. ISI Mode – Access Control Register Assignments

Thirty-two of the 128 control registers are assigned to serve as IACR's and OACR's.

- The control register at addresses 040_8 through 057_8 are assigned as IACR's for input channels 0 through 15 ($0\text{-}17_8$) respectively.
- The control registers at addresses 060_8 through 077_8 are assigned as OACR's for output channels 0 through 15 ($0\text{-}17_8$) respectively.

7.2.3. ISI ACW Format

The format for an ISI ACW includes the G, W, and V fields as shown in the following format:



ISI ACW Format

- V field

The contents of the V field specifies the absolute address of a location in main storage to or from which an I/O word (input, output, or function word) is transferred respectively. After each I/O word is transferred to or from a main storage location, the contents of the V field is either incremented by one, decremented by one, or remains unchanged depending on the contents of the G field.

- W field

The initial contents of the W field specifies the number of I/O words that are to be transmitted between main storage and a subsystem via the associated channel. After each I/O word is transmitted, the contents of the W field is decremented by one. During each I/O word transmission sequence, the contents of the W field is also tested to determine if the terminal condition has been reached. When the terminal condition is detected, the associated channel is deactivated.

- G field

The contents of the G field specifies the following:

(1) G = 00_2

Increment the contents of the V field by one as each I/O word is transmitted between main storage and a subsystem via the associated channel.

(2) G = 10_2

Decrement the contents of the V field by one as each I/O word is transmitted between main storage and a subsystem via the associated channel.

(3) G = 01_2, G= 11_2

The contents of the V field remains unchanged for the duration of the I/O channel activity.

NOTE: The incrementation and decrementation of the contents of the V field are performed in a ones complement subtractive adder. Therefore, the contents of the V field never becomes 777777_8 as a result of the incrementation or decrementation specified by the G field.

7.2.4. ISI ACW Terminal Condition

At the start of the I/O timing chain cycle before an output or function word transfer operation, and at the end of a cycle after an input word transfer operation, the contents of the W field of the associated ACW is tested for the value zero (0) which is the terminal condition. When the terminal condition is detected, the associated channel is deactivated. If the Monitor function is active, the appropriate Monitor Interrupt (ISI Input Monitor, ISI Output Monitor, or ISI Function Monitor Interrupt) also occurs when the terminal condition is detected.

NOTE: Termination for input transfers occurs during the cycle in which the last word is received. For output, an ODR must be presented by the subsystem as if another word will be sent, and it is in this I/O cycle that termination occurs (without a word transfer).

7.2.5. ISI Mode – Input/Output Channel Activity

Two modes of activity are performed on an output channel: the function mode and the output mode. Two modes of activity are also performed on an input channel: the input mode and the external interrupt mode. Each of these four modes of activity are explained thoroughly in 7.2.5.1 through 7.2.5.4.

7.2.5.1. ISI Function Mode

When the LFC or LFCM instruction is executed, an OACW is transferred from main storage location U to the OACR associated with the specified output channel, and the function mode is activated on that channel. If the LFCM instruction is executed, the output monitor function is also activated on that channel.

When a channel is active in function mode, one or more function words are transferred from main storage to a subsystem. The number of function words transmitted depends on what is required by the subsystem to condition it to perform a specific task or operation. The W field of the associated OACW should initially contain the value one (1) if one function word is necessary.

To condition a subsystem such as a magnetic tape subsystem to perform a search operation, it is necessary to transmit two words to the subsystem. In this case, the first word transmitted is the function word and the second word is called an identifier word (ID word). When these words are to be transmitted to a subsystem, they must be located in consecutively addressed locations in main storage; the W field of the associated OACW must contain the value two (2), and the G field of the OACW must specify either incrementation or decrementation of the contents of the V field in order to transfer the function word first and the ID word second.

When the LFC or LFCM instruction is executed, three particular I/O section control circuits (flip-flops) are activated, which are associated with the specified output channel. These circuits are the force external function, external function, and output active control circuits. An output channel is said to be active in function mode when the external function and output active control circuits are simultaneously active.

The force external function control circuit turns on the simulated ODR signal which causes a function word to be forced to a subsystem. Only the first word transmitted to the subsystem after the execution of the LFC or LFCM instruction is forced. The force external function control circuit is deactivated and the simulated ODR is turned off during the I/O timing chain cycle that sequences the transfer of the forced function word to the subsystem.

When a channel is active in function mode, the external function control circuit activates the EF control signal (pulse) as each function word or ID word is transmitted to the subsystem. The external function control circuit is deactivated when an LOC, LOCM, or Disconnect Output Channel (DOC) instruction is executed specifying that channel.

When the output active control circuit for a given channel is in the active state, the I/O section will respond to ODR control signals presented on that channel by a subsystem; it is deactivated when the terminal condition is detected in the OACR for that channel or when a DOC instruction is executed specifying that channel.

If the output channel was activated in function mode by execution of the LFCM instruction, the output monitor control circuit associated with that specified output channel is activated in addition to the three control circuits discussed previously. When the external function control circuit for a given channel is active, the output monitor control circuit allows the ISI Function Monitor Interrupt to occur when the terminal condition is detected in the OACW associated with that channel. In this case, the output monitor control circuit is deactivated when the ISI Function Monitor Interrupt occurs or when an LFC, LOC, or DOC instruction is executed.

The activity performed by the I/O section following the activation of an output channel in function mode is:

- The first function word is forced out with a simulated ODR turned on in the I/O section. The I/O section services the ODR as follows:

  (1) The contents of the associated OACR is tested for the terminal condition. If it is not in the terminal condition, the function word is obtained from the main storage location specified by the contents of the V field of the OACW (see 7.2.6).

  (2) The 36-bit function word and the EF control signal are transmitted on the specified output channel to the subsystem.

  (3) The contents of the OACR are updated. The contents of the W field are decremented. The contents of the V field are incremented, decremented, or left unchanged, as specified by the contents of the G field.

- The subsystem responds to the EF signal by transferring the function word from the output word/function word lines into its control section and turns off the ODR signal if it is on. The function word which is now in the subsystem control section is decoded and the subsystem conditions itself accordingly. After the subsystem has been conditioned it does one of the following:

  (1) Turns on the ODR signal which indicates to the CPU that the subsystem is ready to receive:

    a. another function word;
    b. an ID word;
    c. an output word (7.2.5.2); or,

  (2) Assembles an input word and turns on the IDR control signal (see 7.2.5.3); or,

  (3) Assembles a status word and turns on the EI control signal (see 7.2.5.4).

- When the subsystem turns on an ODR signal in response to the function word it received when the LFC or LFCM instruction was executed, the I/O section responds to the ODR by testing the associated OACW for the terminal condition. One of the following occurs at this point:

  (1) If the contents of the associated OACR are not in the terminal condition and the channel is still active in function mode:

    a. A word (function word or an ID word) is obtained from the main storage location specified by the contents of the V field of the OACW.

    b. The word (function or ID word) and the EF control signal are transmitted on the specified output channel to the subsystem.

    c. The contents of the OACR are updated.

(2) If the contents of the associated OACR are in the terminal condition and the channel is still active in function mode:

a. Function mode on the associated output channel is deactivated.

b. The contents of the OACR are *not* updated.

c. If the monitor function is active on that output channel, the ISI Function Monitor Interrupt occurs (see 7.2.7) and the monitor function is deactivated on that channel.

7.2.5.2. ISI Output Mode

When the LOC or LOCM instruction is executed, an OACW is transferred from instruction storage location U to the OACR associated with the specified output channel, and output mode is activated on that channel. If the LOCM instruction is executed, the output monitor function is also activated on that channel.

When a channel is active in output mode, the subsystem sends ODR control signals to the I/O section via the ODR line associated with that channel. The CPU responds by transferring output words from main storage to the subsystem via the output word/function word lines associated with the same channel. The W field of the associated OACW should initially contain a value that specifies the number of output words to be transmitted.

When the LOC or LOCM instruction is executed, the output active control circuit (same circuit as the one discussed with function mode) associated with the specified output channel is activated. An output channel is said to be active in output mode when the output active control circuit is active and the external function control circuit is simultaneously inactive.

When the output active control circuit is active, it allows the I/O section to respond to ODR control signals presented on that channel by the subsystem; it is deactivated when the terminal condition is detected in the associated OACW or when the DOC instruction is executed specifying that channel.

If the specified output channel was activated by execution of the LOCM instruction, the output monitor control circuit (same circuit as the one discussed with function mode) associated with that output channel is activated in addition. When that channel's external function control circuit *is not* active, the output monitor control circuit allows the ISI Output Monitor Interrupt to occur when the terminal condition is detected. The output monitor control circuit is deactivated when the ISI Output Monitor Interrupt occurs or when an LFC, LOC, or DOC instruction is executed.

The activity performed by the I/O section following activation of the specified output channel in output mode is as follows:

- The I/O section responds to an ODR signal turned on by the subsystem by testing the associated OACW for the terminal condition. At this point one of the following occurs (see 7.2.6 if the OACR is initially in the terminal condition):

  (1) If the contents of OACR are not in the terminal condition:

  a. An output word is obtained from the main storage location specified by the contents of the V field of the OACW.

  b. The output word and the OA signal pulses are transmitted via the output word/function word lines and OA line respectively of the specified output channel to the subsystem.

  c. The contents of the OACR are updated.

  NOTE: The subsystem responds to the OA signal by accepting the output word and turning off the ODR signal. The subsystem then prepares to accept another output word and again turns on the ODR signal and the I/O section again responds by testing the associated OACW for the terminal condition, etc.

  (2) When the contents of the OACR are in the terminal condition:

  a. Output mode on the associated output channel is deactivated.

  b. The contents of the OACR are *not* updated.

  c. If the monitor function is active on that output channel the ISI Output Monitor Interrupt occurs (see 7.2.7) and the monitor function is deactivated on that channel.

7.2.5.3. ISI Input Mode

When the LIC or the LICM instruction is executed, an IACW is transferred from storage location U to the IACR associated with the specified input channel, and input mode is activated on that channel. If the LICM instruction is executed, the input monitor function is also activated on that channel.

When a channel is active in input mode, the subsystem sends IDR control signals to the I/O section via the IDR line associated with that channel. The CPU responds by transferring input words from the subsystem to main storage via the input word/status word lines associated with the same channel. The W field of the associated IACW should initially contain a value that specifies the number of words to be transferred.

When the LIC or LICM instruction is executed, the input active control circuit associated with the specified input channel is activated. An input channel is in input mode when the input active control circuit is active.

When the input active control circuit is active, the I/O section is allowed to respond to IDR's presented on that channel by the subsystem; it is deactivated when the terminal condition is detected or when the DIC instruction is executed specifying that channel.

If the specified input channel was activated by execution of the LICM instruction, the monitor control circuit associated with that input channel is activated in addition to the input active control circuit. This monitor control circuit allows the ISI Input Monitor Interrupt to occur when the terminal condition is detected in the IACW associated with that channel. The monitor control circuit is deactivated when the Input Monitor Interrupt occurs or when a Disconnect Input instruction is executed.

The activity performed by the I/O section following the activation of the specified input channel in input mode is as follows:

- When the subsystem transmits an IDR control signal accompanied by input word signals, the I/O section responds by initiating an input cycle. Then one of the following occurs (see 7.2.6 if the IACR is initially in the terminal condition):

  (1) If the contents of the IACR are *not* in the terminal condition and the contents of the IACW's W field is greater than one (1):

      a. The input word is transferred from the input word/status word lines to the main storage location specified by the contents of the V field of the IACR.

      b. The IA signal is transmitted via the IA line to the subsystem.

      c. The contents of the IACR are updated.

      The subsystem responds to the IA signal by turning off the IDR signal and the current input word signals being held on that input channel. The subsystem then proceeds to prepare another input word for transmission. When ready, the input word signals and the IDR signal are turned on and transmitted to the CPU as described above.

  (2) If the contents of the IACR are *not* in the terminal condition but the contents of the IACW's W field is equal to one (1):

      a. The input word is transferred to the main storage location specified by the contents of the V field of the IACR.

      b. The IA signal is transmitted to the subsystem.

      c. The contents of the IACR are updated (W field is decremented from 1 to 0.

      d. Input mode on the associated input channel is deactivated.

      e. If the input monitor function is active on that input channel, the ISI Input Monitor Interrupt occurs (see 7.2.7) and the input monitor function is deactivated.

7.2.5.4. ISI External Interrupt Mode

A subsystem reports either an abnormal condition or the normal completion of an operation by assembling a status word and then turning on the corresponding status word signals and an EI signal on the input word/status word lines and EI line respectively. The I/O section responds to an EI signal as follows:

- The status word on the input word/status word lines is transferred to a specifically reserved location in main storage. The address of the reserved location is $X00200_8$ for CPU #0, $X00201_8$ for CPU#1, or $X00202_8$ for CPU #2 where X is equal to the contents of MSR.

- The IA signal is transmitted via the IA line of that input channel to the subsystem.

  NOTE: The subsystem responds to the IA signal by turning off the EI and status word signals.

- Through the operation of the I/O priority network, the instruction sequence being performed by the CPU's main control section is altered and the next instruction to be executed is obtained from the ISI External Interrupt fixed address location $X00223_8$ where X is equal to the contents of MSR. Additional ISI External Interrupt information is presented in 8.2.2.

  The I/O section responds to an EI signal regardless of whether or not the I/O channel is active in any mode (input, output or function).

7.2.6. ISI Mode – I/O Channel Activity – ACW Initially in Terminal Condition

- OACW for function mode activity initially in terminal condition. When the LFC or LFCM instruction is executed, the specified output channel is activated in function mode even if the W field of the associated OACW initially contains the value zero. As a result of the execution of the LFC or LFCM instruction, a simulated ODR is turned on for the associated channel. Subsequently, the I/O timing chain is activated as a result of the word transfer request signaled by that ODR, and the terminal condition (W field=0) is detected by the test performed near the start of the I/O timing chain cycle. Upon detecting the terminal condition, function mode on the associated channel is deactivated, the OACW is not updated, and a function word is not transmitted to the subsystem. If the monitor function was activated on the specified output channel (LFCM instruction executed), the Function Monitor Interrupt occurs (see 7.2.7) and the monitor function is deactivated.

- OACW for output mode activity initially in terminal condition. When the LOC or LOCM instruction is executed, the specified output channel is activated in output mode even if the W field of the associated OACW initially contains the value zero. If the ODR signal has already been turned on for the specified output channel, the I/O timing chain is subsequently activated and the terminal condition (W field=0) is detected by the test performed near the start of the I/O timing chain cycle. If an ODR signal is not on when the specified output channel is activated, that channel remains active, but when an ODR signal is turned on for that channel, an I/O timing chain cycle is subsequently activated and the

terminal condition (W field=0) is detected by the test performed near the start of the I/O timing chain cycle. In either case, output mode on the associated channel is deactivated, the OACW is not updated, and an output word is not transferred to the subsystem. If the monitor function was activated on the specified output channel (LOCM executed) the Output Monitor Interrupt occurs (see 7.2.7) and the monitor function is deactivated on that channel.

- IACW for input mode activity initially in terminal condition. The specified input channel is activated in input mode when the LIC or LICM instruction is executed even if the associated IACW's W field contains the value zero. If an IDR signal has already been turned on for the specified input channel, an I/O timing chain cycle is subsequentially activated. Input mode on the channel is deactivated when the terminal condition (W field=0) is detected by the test performed at the start of this I/O timing chain cycle. If an IDR signal is not on when the specified input channel is activated, that channel remains active, but when the IDR signal is turned on, the I/O timing chain cycle is subsequently activated. Again, input mode on the channel is deactivated when the terminal condition (W field=0) is detected by the test performed at the start of this I/O timing chain cycle. In either case, an input word is never transferred to main storage and the contents of the associated IACR remains unchanged. If the monitor function was activated on the specified input channel (LICM executed), the Input Monitor Interrupt occurs (see 7.2.7) and the monitor function is deactivated.

7.2.7. ISI Mode Monitor Interrupts

The monitor interrupt function is activated on a channel operating in ISI mode when that channel is activated by execution of the LICM, LOCM, or LFM instruction. An ISI Input Monitor, ISI Output Monitor or Function Monitor Interrupt occurs as appropriate when the terminal condition is detected in the contents of the associated IACW or OACW respectively. These ISI Monitor Interrupts systematically interrupt the sequence of instructions being executed by the main control section of the CPU's I/O section as explained in 8.5.

The I/O priority network of the CPU determines the priority on which servicing of the ISI Monitor Interrupts is based. Of the three types of ISI Monitor Interrupts, their order of priority from highest to lowest is: Input Monitor Interrupt, Output Monitor Interrupt and Function Monitor Interrupt. If the same type of ISI Monitor Interrupt is simultaneously active on more than one channel, the interrupt on the lowest number channel has the higher priority.

When a Monitor Interrupt occurs, the sequence of instructions being executed by the CPU's main control section is altered and the next instruction to be executed is obtained from a fixed address location in main storage. For the ISI Input Monitor Interrupt, the address of the fixed location is $X00220_8$, for the ISI Output Monitor Interrupt it is $X00221_8$ and for the Function Monitor Interrupt it is $X00222_8$. Additional ISI Monitor Interrupt information is presented in 8.2.1. (In the preceding addresses listed, X is equal to the contents of MSR.)

7.2.8. Programmed Deactivation of an I/O Channel in ISI Mode

If an input channel is active in input mode, execution of the DIC instruction deactivates input mode on the specified input channel. If that channel was activated by execution of the LICM instruction, execution of the DIC instruction also deactivates the monitor function on that input channel.

If an input channel is active in input mode with the monitor function active, execution of the LIC instruction deactivates the monitor function and transfers an ACW from a location specified by the LIC instruction into the IACR for that channel. The input mode remains active on that channel.

If an output channel is active in output mode or function mode, execution of the DOC instruction deactivates output mode or function mode on the specified output channel. If the channel was activated by execution of the LOCM or LFCM instruction, the execution of the DOC instruction also deactivates the monitor function on that output channel.

If an output channel is active in output mode with or without the monitor function active, execution of the LFC or LFCM instruction deactivates output mode. Execution of the LFC instruction also deactivates the monitor function if applicable. In either case, an ACW is transferred from a location specified by the LFC or LFCM instruction to the OACR for that channel and the specified output channel is then activated in function mode with or without the monitor function as specified.

If an output channel is active in the function mode with or without the monitor function active, execution of the LOC or LOCM instruction deactivates the function mode. Execution of the LOC instruction also deactivates the monitor function if applicable. In either case, an ACW is transferred from a location specified by the LOC or LOCM instruction to the OACR for that channel and the specified output channel is then activated in output (data) mode with or without the monitor function as specified.

If an output channel is active in output mode/function mode with the monitor function active, execution of the LOC/LFC instruction deactivates the monitor function and transfers an ACW from a location specified by the LOC/LFC instruction into the OACR for that channel. The output mode/function mode remains active on that channel.

Deactivating the input channel monitor function by execution of the DIC or LIC instruction, or deactivating the output channel monitor function by execution of the DOC, LOC, or LFC instruction does not cause an Input Monitor, Output Monitor, or Function Monitor Interrupt respectively to occur on the specified channel. Due to critical timing conditions, however, there is a remote possibility that the Monitor Interrupt may still occur on a channel even after the deactivating instruction specifying that channel has been executed. The interrupt only occurs if the I/O transfer has just reached completion and the Monitor Interrupt signal is being transmitted to the I/O section at the time when the instruction which deactivated the monitor was executed. In this case, if all

I/O interrupts are in an enabled state, the Monitor Interrupt may still occur within three instruction execution times after the execution of the instruction which deactivated the monitor. If an interrupt with higher priority (a non-I/O interrupt) occurs during the critical timing period, the Monitor Interrupt does not occur.

If all I/O interrupts are in a disabled state when an instruction is executed which disconnects the monitor, it is possible for a Monitor Interrupt which was just being transmitted to the CPU I/O section to be retained. This Monitor Interrupt may occur at a later time when an AAIJ (Allow All I/O Interrupts And Jump) instruction is executed. In this case, the Monitor Interrupt will occur immediately following the first instruction executed after the AAIJ, provided that this instruction is not a PAIJ (Prevent All Interrupts And Jump) instruction. Again, if an interrupt with higher priority (a non-I/O interrupt) occurs during this critical timing period, the Monitor Interrupt will not occur. Further, if an External Interrupt or a Monitor Interrupt occurs on any other I/O channel, the pending Monitor Interrupt will not occur.

The possible situations in which a pending Monitor Interrupt can occur after the execution of an instruction which deactivates the monitor are indicated in the following chart.

If all I/O interrupts are in an enabled state:

Instruction Location	Instruction	
P	Instruction which deactivates the monitor (DIC, LIC, DOC, LOC, or LFC)	
P+1	A	Pending Monitor Interrupt can occur at any of these points
P+2	B	
P+3	C	

If all I/O interrupts are in a disabled state:

Instruction Location	Instruction
N	Instruction which deactivates the monitor (DIC, LIC, DOC, LOC or LFC)
.	.
.	.
.	.
N+XXXX	AAIJ
P	(Not a PAIJ instruction)
P+1	A pending Monitor Interrupt can occur here

7.2.9. Summary of I/O Channel Control Circuit Operation – ISI Mode

Six control circuits are associated with each I/O channel. These six circuits are:

Output Active	(Out Act)
Output Monitor	(Out Mon)
External Function	(EF)
Force External Function	(Force EF)
Input Active	(In Act)
Input Monitor	(In Mon)

The execution of certain I/O instructions affects the control circuits:

INSTRUCTION	EFFECT	
	Activates	Deactivates
LFC	Out Act, EF, Force EF	Out Mon
LFCM	Out Act, Out Mon, EF, Force EF	
LOC	Out Act	Out Mon, EF
LOCM	Out Act, Out Mon	EF
LIC	In Act	In Mon
LICM	In Act, In Mon	
DOC		Out Act, Out Mon, EF
DIC		In Act, In Mon

Some of the I/O section control circuits are affected as a result of the detection of certain conditions in the W field of the associated ACW when the I/O section responds to an ODR or IDR signal. The essential conditions and the effect produced are as follows:

ESSENTIAL CONDITIONS			EFFECT
Responding to:	Control Circuits Active	W field	Deactivates
ODR (simulated)	Out Act, EF, Force EF	≠0	Force EF
ODR (simulated)	Out Act, EF, Force EF	=0	Out Act, Force EF
ODR	Out Act	=0	Out Act
IDR	In Act	1→0	In Act
IDR	In Act	=0	In Act

Monitor Interrupts occur when the I/O section detects certain control circuit conditions. The conditions detected and the effects produced upon their detection are as follows:

ESSENTIAL CONDITIONS		EFFECTS	
Active	Not Active	Interrupt Occurs	Deactivates
In Mon	In Act	ISI Input Monitor Interrupt	In Mon
Out Mon	Out Act, EF	ISI Output Monitor Interrupt	Out Mon
Out Mon, EF	Out Act	ISI Function Monitor Interrupt	Out Mon

7.2.10. Summary of the CPU's Operation Versus the EI, IDR, and ODR Control Signals

The EI control signal is turned on by the subsystem when the subsystem transmits a status word to the CPU. The IDR control signal is turned on by the subsystem when the subsystem transmits an input word to the CPU. The subsystem turns on the ODR signal when it is ready to receive an output word or function word (or ID word) from the CPU.

The I/O section may be active in responding to a control signal from some other I/O channel at the time the particular ODR, IDR, or EI control signal is turned on. Once the I/O section recognizes that this control signal has been turned on, it stores this fact until it reacts to the control signal or until the control signal is turned off, whichever occurs first. The CPU reaction to an ODR signal is to send an OA or EF signal, together with a data word or function word, to the subsystem. Its reaction to an IDR or EI signal is to accept and store a data word or status word and send an IA signal to the subsystem. Once the CPU has reacted to a particular ODR, IDR, or EI control signal, the subsystem must turn off the signal on that line for a minimum of 200 nanoseconds before the CPU I/O section will again recognize that the subsystem is sending another control signal on the same line.

7.2.11. I/O Programming Considerations – ISI Mode

An output channel is conditioned to perform a function word transfer by execution of the LFC or LFCM instruction. To condition an output channel to perform an output word transfer, a combination of the LFC/LFCM and LOC/LOCM instructions is executed. To condition an input channel to perform an input word transfer, a combination of the LFC/LFCM and LIC/LICM instructions is executed.

Other instructions may also be executed in combination with the instructions stated in the preceding paragraph to determine the presence or absence of certain conditions on the specified channel. These instructions provide a program with the means to directly test the condition of a specified channel. These instructions are as follows:

- Jump Function Channel (JFC) (see 6.13.11)

  When a JFC instruction is executed after an LFC or LFCM instruction, the JFC instruction determines whether or not the first function word and EF signals have been transmitted via the specified output channel to the subsystem. Specifically, the JFC instruction tests the condition of the Force EF control circuit. If the Force EF control circuit is active, the program jumps to location U specified by the JFC instruction. If the Force EF control circuit is not active, the program continues with the next consecutive instruction.

  In the CPU's I/O section, the Force EF control circuit is activated when the LFC or LFCM instruction is executed; it is deactivated during the I/O timing chain cycle which controls transmission of the first function word to the respective subsystem following the execution of the LFC or LFCM instruction. If two or more words (function word and ID word, or two or more function words) are to be transmitted to a subsystem, the JFC instruction does not determine whether or not the second, or third, and so forth word has been transmitted to the respective subsystem. The JOC instruction can be used to determine whether or not all of the words have been transmitted to the subsystem.

- Jump Output Channel (JOC) (see 6.13.7)

  The JOC instruction determines whether or not an output channel is active in either function mode or output mode. More specifically, the JOC instruction tests the condition of the output active control circuit. If the output active control circuit is active, the program jumps to location U specified by the instruction. If the output active control circuit is not active, the program continues with the next consecutive instruction. In the I/O section of the CPU the output active control circuit is activated when the LFC, LFCM, LOC or LOCM instruction is executed or when the terminal condition is detected in the associated OACR.

- Jump Input Channel (JIC) (see 6.13.3)

  Execution of the JIC instruction determines whether or not an input channel is active in input mode. Specifically, the JIC instruction tests the condition of the input active control circuit. If the input active control circuit is active, the program jumps to location U specified by the JIC instruction. If the input active control circuit is not active, the program continues with the next consecutive instruction.

  In the CPU's I/O section the input active control circuit is activated when an LIC or LICM instruction is executed; it is deactivated when a DIC instruction is executed or when the terminal condition is detected in the associated IACR.

7.2.11.1. ISI Function Mode Programming Considerations

When a function word transmission is to be performed the specified channel is prepared by activating function mode on that channel. While that channel is active in function mode, one or more function words or a function word and an ID word may be transmitted to the subsystem. Some possible sequences of instruction execution to perform these transfer operations are as follows:

- Sequence #1:

Location	Instruction
N	LFC
N + 1	JFC (Jump to N + 1, continue later to N + 2)
N + 2	xxx

In this sequence, execution of the LFC instruction activates the specified channel in function mode. If the contents of the W field of the associated OACR are initially equal to one or greater, one or more function words are transmitted to the subsystem. Execution of the JFC instruction assures that the program will not proceed to execute the instruction at location N + 2 until the first function word has been transmitted to the subsystem following the execution of the LFC instruction.

- Sequence #2:

Location	Instruction
N	LFC
N + 1	JOC (Jump to N + 1, continue later to N + 2)
N + 2	xxx

In this sequence, execution of the JOC instruction assures that the program will not proceed to execute the instruction at location N + 2 until after function mode is deactivated on that channel. Function mode is deactivated for an I/O channel when the subsystem turns on the ODR signal at a time when function mode is active for that channel but the word count field of the OACW has been decremented to zero.

If the last function word sent to the subsystem calls for input to the CPU rather than output, the subsystem will typically turn on its IDR signal rather than the ODR signal. As a consequence, the output channel will remain active in the function and output mode until the input transfer is completed and the subsystem turns on the ODR signal to indicate that it can accept a new function word.

- Sequence #3:

Location	Instruction	
N	LFCM	
N + 1	xxx	(continue execution)
N + 2	xxx	
M	xxx	(Monitor Interrupt routine
M + 1	xxx	which recognizes completion of the function output sequence)

In this sequence, the program proceeds in normal execution and waits for the Function Monitor Interrupt to occur. After the Function Monitor Interrupt occurs, the program is allowed to proceed.

In all of the above sequences, a programmed timing routine could be used in conjunction with the jump to x function. The timing routine may be used to inform the program when a more than adequate amount of time has elapsed to complete the previously initiated I/O operation. If for some reason the I/O operation has not been completed in the specified period of time, the timing routine informs the program that some error or malfunction exists. This type of routine will prevent the system from being delayed indefinitely waiting for a Monitor Interrupt.

7.2.11.2. ISI Output Mode Programming Considerations

When an output word transmission is to be performed using a channel, two steps are essential to prepare for the operation. The two steps are: (1) The subsystem on that channel must be prepared for output - to turn on ODR signals on the ODR control line and respond to output words transmitted from the CPU's I/O Section and (2) The output mode must be active on the specified channel so that the CPU's I/O section is ready to respond to the ODR's turned on by the subsystem by transferring output words from the locations specified by the OACW. The CPU prepares the subsystem for the output operation by activating the specified channel in function mode and transmitting to the subsystem a particular function word (or words). Execution of the LFC or LFCM instruction activates the channel in function mode. The channel is prepared for the output word transmission by activating the channel in output mode through execution of the LOC or LOCM instruction.

The function word transfers and output data transfers both use the same OACR and output path. Consequently, all function word transfers must be completed before the output data transfers can be initiated. If the LOC or LOCM instruction is executed before the function mode use of the OACR is complete, a situation is created in which the output active and Force EF control circuits are active and the EF control circuit is inactive. Execution of a JFC instruction following the execution of an LFC or LFCM instruction guarantees that this illogical state never occurs.

When a specified channel is active in output (data) mode, it is normally undesirable to disturb that channel by executing an LFC/LFCM or LOC/LOCM instruction specifying that channel until after the output data mode is deactivated. The following three methods can be used to determine whether or not the output (data) mode operation is complete.

(1) A JOC instruction can follow an LOC instruction. The program will jump to the location specified if the output active control circuit is active. The jump location can return the program to repeat the JOC instruction. When the output (data) mode is deactivated, the program will execute the next sequential instruction.

(2) If the output operation was initiated with an LOCM instruction, the Output Monitor Interrupt will occur when the output channel is deactivated.

(3) Some subsystems transmit an EI control signal and a status word to the CPU when the output (data) mode operation is deactivated.

Some possible sequences of instruction execution to prepare to perform an output word transfer operation are as follows:

- Sequence #1:

Location	Instruction
N	LFC (W field of OACW = 1)
N + 1	JFC (Jump to N + 1, continue later to N + 2)
N + 2	LOC or LOCM

This sequence may be used when only one function word is to be transmitted to the subsystem. The LFC instruction activates the specified output channel in function mode which results in the transmission of a function word to the subsystem. The JFC instruction assures that the LOC or LOCM instruction will not be executed until the function mode use of the channel is complete. The LOC or LOCM instruction activates output mode on the specified channel.

- Sequence #2:

Location	Instruction
N	LFC (W field of OACW = 1 or greater)
N + 1	JOC (JUMP to N + 1, continue later to N + 2)
N + 2	LOC or LOCM

This sequence may be used when one or more function words are to be transmitted to the subsystem. In this case the JOC instruction assures that the LOC or LOCM instruction will not be executed until the function mode use of the channel is complete. Specifically, the JOC instruction assures that the LOC or LOCM instruction will not be executed until after function mode is deactivated on the specified channel. In this case the function mode is deactivated when the subsystem turns on the ODR signal after it has received the last function word called for by the OACR.

The operation of the LFC and LOC or LOCM instructions in this sequence are essentially the same as explained in Sequence #1.

- Sequence #3:

Location	Instruction
N	LFCM (W field of OACW = 1 or greater)
N + 1	xxx (continue execution)
N + 2	xxx
M	xxx (Function Monitor Interrupt sends control
M + 1	xxx to this routine.)
M + 2	LOC or LOCM

This sequence could be used when one or more function words are to be transmitted to the subsystem. After execution of the LFCM instruction, the program must not proceed to execute the LOC or LOCM instruction until after the Function Monitor Interrupt occurs on the specified channel. The LFCM instruction conditions the CPU to generate the Function Monitor Interrupt leading to execution of the LOC or LOCM instruction. The LOC or LOCM instruction will not be executed until the function mode use of the specified channel is complete. The function mode is deactivated when the Function Monitor Interrupt occurs.

The operation of the LFCM and LOC or LOCM instructions in this sequence are essentially the same as explained in Sequence #1.

Typically, only one function word is necessary to prepare a subsystem for an output transmission. Subsystems commonly used on an I/O channel operating in ISI mode transmit an external interrupt control signal and status word to the CPU when the output operation is complete. Thus, the program can be informed of completion by the occurrence of the External Interrupt and an Output Monitor Interrupt is unnecessary.

Usually a programmed timing routine is used in conjunction with the sequences illustrated so that the program can be notified when a sufficient time for normal completion of the I/O transfer has elapsed. If for some reason the I/O operation has not been terminated in the specified amount of time, the timing routine informs the program that some error or malfunction exists. This type of routine prevents this system from being delayed indefinitely in the event that some abnormal circumstance prevents the I/O operation from reaching its normal termination.

7.2.11.3. ISI Input Mode Programming Considerations

The two steps essential for an input word transmission are as follows:

(1) The subsystem on the channel must be prepared for input.

(2) The input mode must be active on the specified channel so that the CPU's I/O section is ready to respond to the IDR's turned on by the subsystem. The I/O section of the CPU must accept input word signals from the input word/status word lines and transfer the input words into the main storage locations specified by the IACW. It must also be ready to transmit IA signals through the IA line to the subsystem.

The subsystem is prepared for the input word transmission by activating the specified channel in function mode and transmitting a particular function word (or words) to the subsystem. Execution of the LFC or LFCM instruction activates the specified channel in function mode. The specified input channel is prepared for the input transmission by activating the channel in input mode through execution of the LIC or LICM instruction.

No conflict occurs when preparing an I/O channel for input data transfer because the LFC uses the output path and OACR associated with that channel and the LIC or LICM initiates the input path and associated IACR.

In the input transmission of many systems, the CPU's I/O section must accept the input words at a rate governed by the subsystem. If the CPU has not accepted a particular input word from the subsystem by the time the subsystem is ready to transmit the next input word, a situation arises which is described as data pile up. When data pile up occurs, the subsystem stops the input transmission and transmits an EI signal and status word to the CPU's I/O section reporting the data pile up condition.

In initiating input, if the LFC/LFCM instruction is executed first, followed some time later by execution of the LIC/LICM instruction, it is possible that before the channel is active in input mode the subsystem (such as a magnetic tape subsystem) could already have data pile up. This cause of data pile up can be avoided by executing the LIC/LICM instruction first, followed some time later by execution of the LFC/LFCM instruction. Some suggested sequences of instruction execution to perform an input word transfer operation are as follows:

- Sequence #1:

Location	Instruction
N	LIC or LICM
N + 1	LFC (W field of OACW = 1 or more)
N + 2	JFC (Jump to N + 2 continue later to N + 3)
N + 3	xxx

In this sequence, the input channel is first activated in input mode and then function mode is activated on the output channel. As soon as the subsystem is prepared to perform the input operation, the input channel is already prepared and no data pile up should occur. The use of the JFC instruction assures the program that the first (if W field > 1 initially) function word has been transmitted to the subsystem.

Once the specified channel is active in input mode, it is normally undesirable to disturb the channel by executing an LFC/LFCM or LIC/LICM instruction specifying that channel until after the input mode operation is completed. A program can determine whether or not the input operation is complete through execution of the JIC instruction, or by waiting for the Input Monitor Interrupt to occur on the channel if it was activated with the LICM instruction, or by waiting for an External Interrupt to occur on the channel.

- Sequence #2

Location	Instruction
N	LIC or LICM
N + 1	LFC (W field of OACW = 1 or more)
N + 2	JOC (Jump to N + 2, continue later to N + 3)
N + 3	xxx

This sequence is like Sequence #1 except that the JOC instruction will not allow the program to proceed to instruction N + 3 until after the subsystem has completed the input word transmission operation and has returned to the at rest state. In this case, after a function word or words is/are transmitted to the subsystem, the subsystem prepares for an input operation. When the subsystem is ready, it turns on the IDR signal, and the transmission of input words begins. The subsystem responds with an IDR signal and not an ODR signal after it receives the function word (or words if the W field > 1 initially). Therefore the output active control circuit remains active and the JOC instruction does not allow the program to proceed to the instruction in location N + 3. But when the input operation is complete and the subsystem returns to the idle state, it turns on the ODR control signal, and the CPU's I/O section responds by deactivating the output active control circuit. After the output active control circuit is deactivated, the JOC instruction allows the program to proceed to execute the instruction at location N + 3.

- Sequence #3

Location	Instruction
N	LFC (W field of OACW = 1 or more)
N + 1	JFC (Jump to N + 1 continue later to N + 2)
N + 2	LIC or LICM

This sequence is permissible if the speed of execution is sufficient to prevent data pile up (magnetic drum subsystem). However, if the JOC instruction was used instead of the JFC instruction the program would stall on the output active test and never be allowed to proceed to execute the instruction at location N + 2 (see 7.2.11.2).

7.3. ESI MODE – I/O OPERATION

The ESI mode of I/O channel operation is used on a channel which is assigned a special ESI subsystem. Many special peripheral devices (ESI devices) or communications lines can be assigned to one ESI subsystem. An example of an ESI subsystem is the Communication Terminal Module Control (CTMC) subsystem. An example of an ESI device is the Communications Line Terminal (CLT).

When an I/O channel is active and operating in ESI mode (see 7.1.4), either half words (18 bits) or quarter words (9 bits) are transferred to or from main storage (word serial, bit parallel) via the CPU I/O section and I/O channel from or to an ESI subsystem. The ESI subsystem in turn transmits these words to, or receives these words from, the ESI devices. The aggregate of transmissions to or from all of the ESI devices associated with an ESI subsystem are performed on a time sharing basis (multiplexed) via the ESI subsystem and I/O channel.

7.3.1. General Description – Operation of an I/O Channel in ESI Mode

A channel operating in ESI mode is activated when any of the following instructions are performed:

Load Input Channel (LIC)
Load Input Channel and Monitor (LICM)
Load Output Channel (LOC)
Load Output Channel and Monitor (LOCM)

The particular channel activated is specified by the logical sum of the contents of the a field of the instructions and the contents of the Channel Select Register (a **OR** CSR). The LFC and LFCM instructions are legitimate instructions for transmitting function words in ESI mode, as explained in 7.3.5.1. However, they are not included in the list above because their execution, when specifying a channel in ESI mode, does not cause the channel to be activated in function mode; that is, the output active control circuit or the output active and output monitor control circuits are never activated as a result of execution of the LFC or LFCM instruction respectively.

When the LIC or LICM instruction is executed (specifying a channel that is operating in ESI mode), the specified input channel is activated in input mode; when the LICM instruction is executed, the input monitor function is also activated for that input channel. When the LOC or LOCM instruction is executed, the specified output channel is activated in output mode; when the LOCM instruction is executed, the output monitor function is also activated for that output channel. When any of the instructions LIC, LICM, LOC, or LOCM is executed, an ACW is *not* transferred from a location specified by the instruction to an IACR or OACR for the associated channel as is the case when a channel is operating in ISI mode.

The transmissions of input words and output words to or from the ESI subsystem are accompanied by an ESI value (15 bits) from the subsystem. The ESI value is augmented with the value in MSR to form the absolute address of a main storage location containing an ESI ACW.

One ESI ACW is usually associated with each ESI device. The contents of the ESI ACW specifies the following:

- absolute address of a location in main storage to or from which an input word or output word is transferred;
- number of words to be transferred;
- whether the address portion is to be incremented, decremented or left unchanged;
- in ESI quarter word output whether or not the channel is deactivated when the terminal condition is detected.

Since the address of the ACW is specified by a source which is basically external with respect to the CPU or main storage, the term Externally Specified Index (ESI) is used to denote this particular mode of I/O operation.

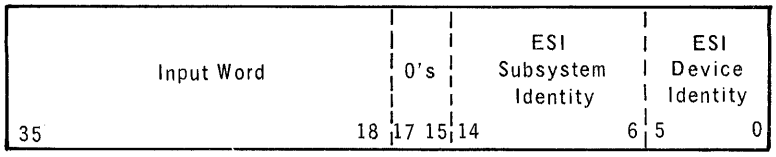
Deactivating of an input or output channel operating in ESI mode occurs when the DIC or DOC instruction respectively is executed. Deactivation of an input channel also occurs when the terminal condition is detected in an associated ESI ACW. Deactivation of an output channel normally occurs when the terminal condition is detected in an associated ESI ACW; in some specified cases in quarter word operation, the output channel is not deactivated when the terminal condition in the ESI ACW is detected.

7.3.2. ESI Word Format

As previously mentioned, each IDR and ODR control signal transmitted to the CPU I/O section by the ESI subsystem is also accompanied by an ESI value. In each case, the ESI value is transmitted via the input word/status word lines. The ESI value is augmented in the I/O section by the value in Memory Select Register (MSR) to form the absolute address of a main storage location containing an ESI ACW.

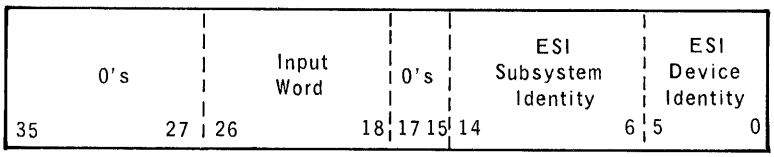
The ESI format is the same for both an input or output operation. For an input operation, however, the IDR is accompanied by both the ESI value and input data.

For an ESI half word input operation, the information transmitted to the CPU I/O section on the input word/status word lines when the IDR control signal is on has the following format:



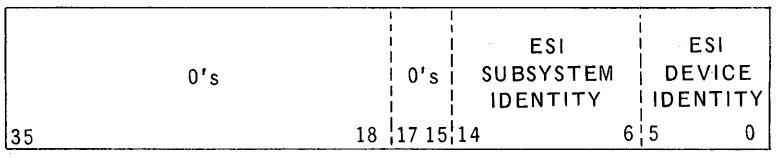
ESI Word Format – Half Word Input

For an ESI quarter word input operation, the information transmitted to the CPU I/O section on the input word/status word lines when the IDR control signal is on has the following format:



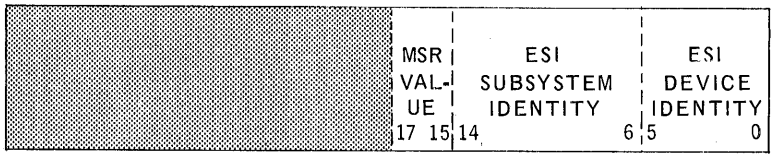
ESI Word Format – Quarter Word Input

For an ESI half word or quarter word output operation, the information transmitted to the CPU's I/O section on the input word/status word lines when the ODR control signal is on has the following format:



ESI Word Format – Output

After the ESI word is accepted by the I/O section, it is augmented with the value from MSR. The biased ESI address is an 18-bit value which specifies the absolute address of a location in main storage. The biased ESI address has the following format:



Biased ESI Address Format

As each input word is transferred to main storage, in an ESI input operation, the biased ESI value is stored in bits 35 through 18 of the IACR associated with the channel. As each output word is transferred from main storage in an ESI output operation, the biased ESI value is stored in bits 17 through 00 of the IACR associated with the channel.

The subsystem and device identity fields allow a unique ESI ACW in main storage to be associated with each ESI device. Each unique ESI ACW controls the address in main storage to which, or from which, an input word or output word is transferred that has been received from, or is being transmitted to, a particular ESI device.

7.3.3. ESI ACW Formats

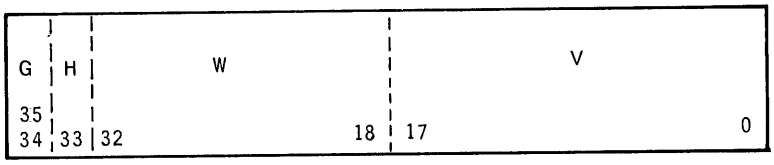
One ESI ACW format is used for ESI half word input and output operations. A second ESI ACW format is used for ESI quarter word input and output operations. An I/O channel is implemented for either half word or quarter word operation through the use of a particular plug-in card.

For each ESI half word operation, two I/O words are stored in each main storage location specified by the ESI half word ACW.

When the ESI quarter word operation is in effect, four I/O words are stored in each main storage location specified by the ESI quarter word ACW.

7.3.3.1. ESI Half Word ACW Format

The ESI half word ACW includes four fields (G, H, W, and V) and has the following format:



ESI Half Word ACW Format

The four fields of the ESI half word ACW are defined as follows:

- V field

  The contents of the V field (bits 17 through 0) specifies the absolute address of a location in main storage to or from which a half word ESI I/O word is transferred. As each I/O word is transferred to or from a main storage location, the contents of the V field is either incremented by one, decremented by one, or remains unchanged depending on the contents of the G and H fields.

- W field

  The initial contents of the W field (bits 32 through 18) specifies the number of half word ESI I/O words that are to be transmitted between main storage and an ESI device via the associated channel. As each I/O word is transmitted, the contents of the W field is decremented by one and tested to determine if the terminal condition has been reached.

- H field

  When the CPU I/O section is responding to an IDR or ODR, the contents of the H field (bit 33) specifies the following:

  (1) H = 0

  Transfer the ESI I/O word to or from bits 17 through 0 of the main storage location addressed by the contents of the V field and change the contents of the H field from 0 to 1.

  (2) H = 1

  Transfer the ESI I/O word to or from bits 35 through 18 of the main storage location addressed by the contents of the V field and change the contents of the H field from 1 to 0.

- G field

The contents of the G field (bits 35 and 34) specifies the following:

(1) G = 00_2

Increment the content of the V field by one when the contents of the H field is changed from 1 to 0.

(2) G = 10_2

Decrement the content of the V field by one when the contents of the H field is changed from 1 to 0.

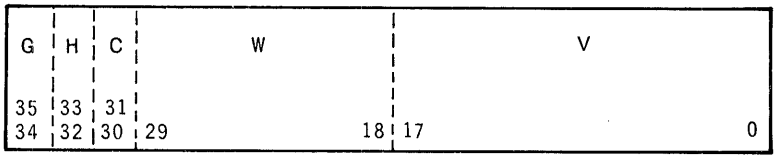
(3) G = 01_2 or 11_2

The contents of the V field remains unchanged when the content of the H field is changed from 1 to 0.

NOTE: The incrementation and decrementation of the contents of the V field are performed in a ones complement subtractive adder. Therefore, the contents of the V field can never become 777777_8 as a result of the incrementation or decrementation specified by the G field.

7.3.3.2. ESI Quarter Word ACW Format

The ESI quarter word ACW includes five fields (G, H, C, W and V) and has the following format:



ESI Quarter Word ACW Format

The five fields of the ESI quarter word ACW are defined as follows:

- V field

The contents of the V field (bits 17 through 0) specifies the absolute address of a location in main storage to or from which a quarter word ESI I/O word is transferred. As each I/O word is transferred to or from a main storage location, the content of the V field is either incremented by one, decremented by one, or remains unchanged depending on the contents of the G and H fields.

- W field

The initial contents of the W field (bits 29 through 18) specifies the number of quarter word ESI I/O words that are to be transmitted between main storage and an ESI device via the associated channel. As each input word is transmitted, the content of the W field is decremented by one and tested to determine if the terminal condition has been reached. For output word transmission, the contents of the W and C fields combined are tested to determine if the terminal condition has been reached.

- C field

The contents of the C field (bits 31 and 30) specifies a special action to be performed by the I/O section when the W field is decremented from 2 to 1 during an output word transmission. Bit 31 of the C field is called the Early EOT Control bit (EOT = end of transmission); bit 30 is called the Early Monitor and Deactivate Control bit.

(1) $C = 00_2$

No special action is performed when the contents of the W field of the ACW addressed by the biased ESI word is decremented from 2 to 1.

(2) $C = 01_2$ (Early Monitor and Deactivate Control Bit set)

When the CPU I/O section responds to an ODR control signal and ESI word signals, and the contents of the W field of the ACW addressed by the biased ESI word is a 2, the operation of the I/O section is as follows:

a. A quarter word ESI output word is transferred from the main storage location specified by the V field of the ACW and is transmitted via the associated output channel with an OA signal to the ESI subsystem.

b. The ACW is updated which includes decrementation of the contents of the W field from 2 to 1.

c. The output channel is deactivated. An ESI Output Monitor Interrupt occurs if the output monitor function is active on that channel.

(3) $C = 10_2$ (Early EOT Control Bit set)

When the CPU I/O section responds to an ODR control signal and ESI word signals, and the contents of the W field of the ACW addressed by the biased ESI word is 2, the operation of the I/O section is as follows:

a. A quarter word ESI output word is transferred from the main storage location specified by the V field of the ACW; this word plus an EOT bit (a 1 bit in bit 9 for a total of 10 bits) are transmitted via the associated output channel with an OA signal to the ESI subsystem. (The EOT bit causes the ESI device to discontinue operation, that is, turns off.)

b. The ACW is updated (includes decrementing the contents of the W field from 2 to 1).

c. The output channel is not deactivated and an ESI Output Monitor Interrupt does not occur.

(4) C = 11_2 (Early EOT and Monitor and Deactivate Bits set)

When the CPU I/O section responds to an ODR control signal and ESI word signals, and the contents of the W field of the ACW addressed by the biased ESI word is 2, the operation of the I/O section is as follows:

a. A quarter word ESI output word is transferred from the main storage location specified by the V field of the ACW; this word plus an EOT bit (a 1 bit in bit 9 for a total of 10 bits) are transmitted via the associated output channel with an OA signal to the ESI subsystem. (The EOT bit causes the ESI device to discontinue operation, that is, turns off.)

b. The ACW is updated (includes decrementing the contents of the W field from 2 to 1).

c. The output channel is deactivated. An ESI Output Monitor Interrupt occurs if the output monitor function is active on that channel.

NOTE: The contents of the C field are ignored when a quarter word ESI input word transmission operation is being performed.

- H field

When the CPU I/O section is responding to an IDR or ODR signal, the contents of the H field (bits 33 and 32) specifies the following:

(1) H = 00_2

Transfer the ESI I/O word to or from bits 35 through 27 of the main storage location addressed by the contents of the V field and change the contents of the H field from 00_2 to 01_2.

(2) H = 01_2

Transfer the ESI I/O word to or from bits 26 through 18 of the main storage location addressed by the contents of the V field and change the contents of the H field from 01_2 to 10_2.

(3) H = 10_2

Transfer the ESI I/O word to or from bits 17 through 9 of the main storage location addressed by the contents of the V field and change the contents of the H field from 10_2 to 11_2.

(4) H = 11_2

Transfer the ESI I/O word to or from bits 8 through 0 of the main storage location addressed by the contents of the V field and change the contents of the H field from 11_2 to 00_2.

- G field

The contents of the G field (bits 35 and 34) specifies the following:

(1) G = 00_2

Increment the contents of the V field by one when the contents of the H field is changed from 11_2 to 00_2.

(2) G = 10_2

Decrement the contents of the V field by one when the contents of the H field is changed from 11_2 to 00_2.

(3) G = 01_2 or 11_2

The contents of the V field remain unchanged when the contents of the H field is changed from 11_2 to 00_2.

NOTE: The incrementation and decrementation of the contents of the V field are performed in a ones complement subtractive adder. Therefore, the contents of the V field can never become 777777_8 as a result of the incrementation or decrementation specified by the G field.

7.3.4. ESI ACW Terminal Condition Detection

The terminal condition is detected when the W field of the ACW addressed by the associated biased ESI word is decremented from 1 to 0 during any ESI half word or quarter word input or output transmission operation.

- Terminal Condition Detection – ESI Output

When the I/O section responds to an ODR control signal and ESI word signals and the contents of the W field of the ACW addressed by the biased ESI word is 1, the operation of the I/O section is as follows:

(1) An ESI output word is transferred from the main storage location specified by the V field of the ACW and is transmitted via the associated output channel with an OA signal to the ESI subsystem.

(2) The ACW is updated (includes decrementing the contents of the W field from 1 to 0).

(3) The terminal condition is detected upon the decrementation of the W field from 1 to 0; as a result the following occurs:

a. The output channel is deactivated.

b. An ESI Output Monitor Interrupt occurs if the output monitor function is active on that channel.

- Terminal Condition Detection – ESI Input

When the I/O section responds to an IDR control signal, and ESI word signals, and the contents of the W field of the ACW addressed by the biased ESI word is 1, the operation of the I/O section is as follows:

(1) The ESI input word is transferred into the main storage location addressed by the V field of the ACW and an IA signal is transmitted via the associated input channel to the ESI subsystem.

(2) The ACW is updated (includes decrementing the contents of the W field from 1 to 0).

(3) The terminal condition is detected upon the decrementation of the W field from 1 to 0; as a result the following occurs:

a. The input channel is deactivated.

b. An ESI Input Monitor Interrupt occurs if the input monitor function is active on that channel.

7.3.5. ESI Mode – Input/Output Channel Activity

Four different types of activity can be performed on an I/O channel that is operating in ESI mode. Essentially, these activities consist of: function word transmissions, output word transmissions, input word transmissions, and external interrupt status word transmissions. Each of these activities are explained in the following sections.

7.3.5.1. ESI Mode – Function Word Transmission Activity

When the LFC or LFCM instruction is executed, an OACW is transferred from the location specified by the instruction to the OACR associated with the specified output channel. The OACW conforms to the ISI - OACW format.

After the LFC or LFCM instruction is executed, one function word is transferred from the main storage location specified by the OACW and is transmitted via the output word/function word lines of the associated channel to an ESI subsystem. Only one function word is transmitted via a channel operating in ESI mode to an ESI subsystem per each execution of the LFC or LFCM instruction. The specified channel is never activated in function mode; and execution of the LFCM instruction never activates the monitor function on the specified output channel. The activity performed by the I/O section following the execution of the LFC or LFCM instruction is as follows:

- The force external function (Force EF) control signal is activated and causes a simulated ODR to be turned on in the I/O section. The I/O section in turn services the ODR as follows:

  (1) The contents of the associated OACR is tested for the terminal condition. If it is not in the terminal condition (W field $\neq 0$), the function word is obtained from the main storage location specified by the contents of the V field of the OACW.

  (2) The function word (up to 36 bits in length) and EF control signal pulses are transmitted via the output word/function word lines and the EF line respectively of the specified output channel to the ESI subsystem.

  (3) The contents of the OACR are updated. (The contents of the W field are decremented. The contents of the V field are either incremented, decremented or left unchanged as specified by the contents of the G field.)

NOTE: If the contents of the OACR is initially in the terminal condition (W field = 0), no function word or EF signals are transmitted and the contents of the OACR are not updated. When the LFC or LFCM instruction is executed, two particular I/O section control circuits (flip-flops) which are associated with the specified output channel are activated. These circuits are the Force EF and the EF control circuits.

The Force EF control circuit turns on the simulated ODR signal which causes a function word to be forced to the ESI subsystem. The EF control circuit activates the EF control signal (pulse) as the function word is transmitted to the ESI subsystem. Both the Force EF and the EF control circuit are deactivated during the I/O timing chain cycle that sequences the transmission of the forced function word to the ESI subsystem. If the W field of the OACW initially contains the value 0, the Force EF and EF control circuits are deactivated even though no function word transmission occurs.

When a specified I/O channel is operating in ESI mode, both the input and output channels may be simultaneously active in input mode and output mode respectively. The input mode or output mode operation of that I/O channel is not affected by the occurrence of a function word transmission.

7.3.5.2. ESI Output Mode

When the LOC or LOCM instruction is executed, the output channel specified by the LOC/LOCM instruction is activated in output mode. If the LOCM instruction is executed, the output monitor function is also activated on that channel. In ESI mode, an OACW is not transferred from a location specified by the instruction to the associated OACR as is the case when a channel is operating in ISI mode.

When an output channel, which is operating in ESI mode, is active in output mode, output words are transferred from main storage and are transmitted to an ESI subsystem via the output word/function word lines for that channel. This operation is in response to ODR control signals and ESI word signals presented to the CPU's I/O section by the ESI subsystem via the ODR line and input word/status word lines respectively.

The activity performed by the I/O section in response to an ODR signal and ESI word signals on an active ESI output channel is as follows:

- When an ESI subsystem turns on an ODR signal, the CPU's I/O section responds by augmenting the ESI value with the contents of the MSR and transferring the resulting biased ESI address into bits 17 through 0 of the IACR associated with that channel.

- An ESI ACW is transferred to the I/O section from the main storage location specified by the biased ESI address.

- The ESI ACW is then tested for the terminal condition.

  (1) If the contents of the ESI ACW is not in the terminal condition:

    (a) An ESI output word is obtained from the specified portion of the main storage location addressed by the V field of that ESI ACW.

    (b) The ESI output word and OA signal pulses are transmitted via the output word/function word lines and OA line respectively to the ESI subsystem. For an ESI half word output operation, the output word is transmitted via lines 17 through 0. For an ESI quarter word output operation, the output word is transmitted via lines 8 through 0.

      NOTE: The ESI subsystem responds to the OA signal by accepting the ESI output word and turning off the ODR signal and ESI word signals.

    (c) The contents of the ESI ACW are updated. That is, the contents of the W field is decremented by one. The contents of the V field may or may not be incremented or decremented by one depending on the contents of the G and H fields, and the contents of the H field are changed (see 7.3.3.1 and 7.3.3.2).

    (d) The updated ESI ACW is tested again for the terminal condition and is also transferred into the main storage location specified by the biased ESI address. If at this time the ESI ACW is in the terminal condition the following occurs:

      – Output mode on the associated output channel is deactivated.

      – If the output monitor function is active on that output channel, the ESI Output Monitor Interrupt occurs (see 7.3.6).

(2) See 7.3.6 for an explanation of the operation of the CPU's I/O section if the ESI ACW is initially in the terminal condition when the I/O section responds to an ODR signal.

NOTE: When the LOC or LOCM instruction is executed, the OA control circuit associated with the specified output channel is activated. An ESI output channel is said to be active in output mode when the OA control circuit is active.

When the OA control circuit is active on a channel operating in ESI mode, it allows the CPU's I/O section to respond to ODR control signals and ESI word signals presented on that I/O channel by an ESI subsystem; it is deactivated when the terminal condition is detected in an associated ESI ACW or when the DOC instruction is executed specifying that channel. If the specified output channel was activated in output mode by execution of the LOCM instruction, the output monitor control circuit associated with that output channel is activated in addition to the OA control circuit. The output monitor control circuit allows the ESI Output Monitor Interrupt to occur when the terminal condition is detected in an ESI ACW associated with that channel. When a channel is operating in ESI mode, the output monitor control circuit is deactivated when the ESI Output Monitor Interrupt occurs, or when the LOC or DOC instruction is executed.

7.3.5.3. ESI Input Mode

When the LIC or LICM instruction is executed, the input channel specified by the LIC/LICM instruction is activated in input mode. If the LICM instruction is executed, the input monitor function is also activated on that channel. In ESI modes, an IACW is not transferred from a location specified by the instruction to the associated IACR as is the case when a channel is operating in ISI mode.

In ESI input mode, the ESI subsystem transmits IDR control signals and ESI word signals to the CPU's I/O section via the IDR line and input word/status word lines respectively. The I/O section responds by transferring into main storage the input words that are also transmitted from the ESI subsystem via the input word/status word lines of the same channel (see 7.3.2).

The activity performed by the I/O section in response to an IDR signal and ESI word signals on an active ESI input channel is as follows:

- When an IDR signal has been turned on, the CPU's I/O section responds by augmenting the ESI word with the contents of the MSR and transferring the resulting biased ESI address into bits 35 through 18 of the related IACR.
- An ESI ACW is transferred to the I/O section from the main storage location specified by the biased ESI address.
- The ESI ACW is then tested for the terminal condition.

  (1) If the contents of the ESI ACW is not in the terminal condition:

a. The ESI input word is transferred from the input word/status word lines to the specified portion of the main storage location addressed by the V field of that ESI ACW. For an ESI half word input operation the input word is transferred from lines 35 through 18. For an ESI quarter word input operation, the input word is transferred from lines 26 through 18.

b. The IA signal is transmitted via the IA line of the input channel to the ESI subsystem.

NOTE: The ESI subsystem responds to the IA signal by turning off the IDR, ESI word, and input word signals.

c. The contents of the ESI ACW are updated. That is, the contents of the V field may or may not be incremented or decremented by one depending on the contents of the G and H fields; and the contents of the H field are changed (see 7.3.3.1 and 7.3.3.2).

d. The updated ESI ACW is tested again for the terminal condition and is also transferred into the biased ESI address specified main storage location. If at this time the ESI ACW is in the terminal condition the following occurs:

- Input mode on the associated input channel is deactivated.

- If the input monitor function is active on that input channel, the ESI Input Monitor Interrupt occurs (see 7.3.6).

(2) See 7.3.6. for an explanation of the operation of the CPU's I/O section if the ESI ACW is initially in the terminal condition when the I/O section responds to an ODR signal.

NOTE: When the LIC or LICM instruction is executed, the input active control circuit associated with the specified input channel is activated. An ESI input channel is said to be active in input mode when the input active control circuit is active.

When the input active control circuit is active on a channel operating in ESI mode, it allows the CPU's I/O section to respond to IDR control signals and ESI word signals and input word signals presented on that input channel by an ESI subsystem; it is deactivated when the terminal condition is detected in an associated ESI ACW or when the DIC instruction is executed specifying that channel. If the specified input channel was activated in input mode by execution of the LICM instruction, the input monitor control circuit associated with that input channel is activated in addition to the input active control circuit. The input monitor control circuit allows the ESI Input Monitor Interrupt to occur when the terminal condition is detected in an ESI ACW associated with that channel. When a channel is operating in ESI mode the input monitor control circuit is deactivated when the ESI Input Monitor Interrupt occurs or when the LIC or DIC instruction is executed.

7.3.5.4. ESI External Interrupt Mode

An ESI subsystem reports either an abnormal condition or the normal completion of an operation by assembling a status word and then turning on corresponding status word signals and an EI signal on the input word/status word lines and EI line respectively. The CPU's I/O section responds to an EI signal on a channel in ESI mode as follows:

- The status word on the input word/status word lines is transferred to a specifically reserved location in main storage. The address of the reserved location is $X00200_8$ for CPU#0, $X00201_8$ for CPU#1, or $X00202_8$ for CPU#2, where X is equal to the contents of the MSR.

- The I/O section transmits an IA signal (pulse) via the IA line of that input channel to the ESI subsystem.

  NOTE: The ESI subsystem responds to the IA signal by turning off the EI and status word signals.

- Through the operation of the I/O priority network, the sequence of instruction execution being performed by the CPU's main control section is altered and the next instruction to be executed is obtained from the ESI EI fixed address location $X00227_8$ where X is equal to the contents of the MSR. Additional ESI EI information is presented in 8.2.2.

The CPU's I/O section responds to an EI signal regardless of whether or not the I/O channel is active in any mode (input or output mode).

7.3.6. ESI Mode – I/O Channel Activity – ACW Initially in Terminal Condition

- OACW for function word transmission initially in the terminal condition.

  When the LFC or LFCM instruction is executed, the Force EF and EF control circuits are activated even if the associated OACW is in the terminal condition (W field = 0). As a result of the execution of the LFC or LFCM instruction, a simulated ODR is turned on for the associated channel. Subsequently, the I/O timing chain is activated as a result of the word transfer request signaled by that ODR, and the terminal condition (W field = 0) is detected by the test performed near the start of the I/O timing chain cycle. When the terminal condition is detected, the Force EF and EF control circuits are deactivated, the OACW is not updated, and a function word is not transmitted to the subsystem.

- ESI ACW for output word transmission activity initially in the terminal condition.

  If the W field of the ACW addressed by the associated biased ESI word is already in the terminal condition (W field = 0) when the CPU's I/O section responds to an ODR and ESI word signals, the operation of the I/O section is as follows:

  (1) A half word (18 bits) of all 1 bits is transmitted via the function word/output word lines (17 through 0) of the associated output channel and an OA signal is transmitted via the OA line of that channel to the ESI subsystem. This word is called an EOT word (end of transmission).

(2) The ESI ACW is not updated.

(3) The output channel is not deactivated. An ESI Output Monitor Interrupt does not occur.

- ESI ACW for input word transmission activity initially in the terminal condition.

  If the W field of the ACW addressed by the associated biased ESI word is already in the terminal condition (W field = 0) when the CPU's I/O section responds to an IDR and ESI word signals, the operation of the I/O section is as follows:

  (1) The input word is not transferred into main storage, but an IA signal is transmitted via the IA line of the associated input channel to the ESI subsystem.

  (2) The ESI ACW is not updated.

  (3) The input channel is not deactivated. An ESI Input Monitor Interrupt does not occur.

7.3.7. ESI Mode Monitor Interrupts

The monitor function is activated on an input channel operating in ESI mode when that channel is activated by execution of the LICM instruction; the monitor function is activated on an output channel operating in ESI mode when that channel is activated by execution of the LOCM instruction. An ESI Input Monitor or ESI Output Monitor Interrupt occurs as appropriate when the terminal condition is detected in the contents of the associated ESI ACW. These ESI monitor interrupts systematically interrupt the sequence of the instructions being executed by the main control section of the CPU's I/O section as explained in 8.2.1.4 and 8.2.1.5.

The I/O priority network of the CPU determines the priority on which servicing of the ESI monitor interrupts is based. The ESI Input Monitor Interrupt has higher priority than the ESI Output Monitor Interrupt. If the same type of ESI monitor interrupt is simultaneously active on more than one channel, the interrupt on the lowest number channel has the higher priority (see 8.2.5 for interrupt priority).

When a monitor interrupt occurs, the sequence of instructions being executed by the CPU's main control section is altered and the next instruction to be executed is obtained from a fixed address location in main storage. For an ESI Input Monitor Interrupt, the address of the fixed location is $X00224_8$, and for the ESI Output Monitor Interrupt, it is $X00225_8$ (X is equal to the contents of the MSR). Additional ESI monitor interrupt information is presented in 8.2.1.

In some special cases in ESI quarter word output operation, the ESI Output Monitor Interrupt occurs at a time other than when the terminal condition is detected in the contents of the associated ESI ACW. In these cases, the ESI Output Monitor Interrupt occurs when the W field of the ESI ACW is decremented from 2 to 1 and the contents of the C field of that quarter word ESI ACW is 01_2 or 11_2 provided of course that the monitor function is active on that output channel.

A Function Monitor Interrupt will never occur on a channel operating in ESI mode.

7.3.8. Programmed Deactivation of an I/O Channel in ESI Mode

If an ESI input channel is active in input mode, execution of the DIC instruction specifying that channel deactivates input mode on that channel; if that input channel was activated by execution of the LICM instruction, execution of the DIC instruction also deactivates the ESI input monitor function on that channel. If an ESI input channel is active in input mode with the monitor function active, execution of the LIC instruction deactivates the monitor function but the input mode remains active on that channel.

If an ESI output channel is active in output mode, execution of the DOC instruction specifying that channel deactivates output mode on that channel; if that output channel was activated by execution of the LOCM instruction, execution of the DOC instruction also deactivates the ESI output monitor function on that channel. If an ESI output channel is active in output mode with the monitor function active, execution of the LOC instruction deactivates the monitor function, but the output mode remains active on that channel.

Deactivating the ESI input monitor function by execution of the DIC or LIC instruction, or deactivating the ESI output monitor function by execution of the DOC or LOC instruction does not cause an ESI Input Monitor or ESI Output Monitor Interrupt respectively to occur on the channel specified by the DIC, LIC, DOC, or LOC instruction. But, due to critical timing conditions, there is a remote possibility that the monitor interrupt may still occur on a channel even after the deactivating instruction specifying that channel has been executed.

The interrupt will only occur if the I/O transfer had just reached completion and the monitor interrupt signal was being transmitted to the CPU's I/O section at the time the DIC, LIC, DOC, or LOC instruction was executed. The details of this critical timing condition are the same for ESI mode as for ISI mode and are explained in 7.2.8.

7.3.9. Summary of I/O Channel Control Circuit Operation – ESI Mode

Six control circuits are associated with each I/O channel. These six circuits are as follows:

External Function	(EF)
Force External Function	(Force EF)
Output Active	(Out Act)
Output Monitor	(Out Mon)
Input Active	(In Act)
Input Monitor	(In Mon)

The execution of certain I/O instructions affects the control circuits. The instructions and the effect they produce are as follows:

INSTRUCTION	EFFECT	
	Activates	Deactivates
LFC	EF, Force EF	
LFCM	EF, Force EF	
LOC	Out Act	Out Mon
LOCM	Out Act, Out Mon	
LIC	In Act	In Mon
LICM	In Act, In Mon	
DOC		Out Act, Out Mon, EF
DIC		In Act, In Mon

In the operation of the I/O section of the CPU, some of the control circuits are affected as a result of the detection of certain conditions in the W field of the associated ESI ACW when the CPU's I/O section responds to an ODR or IDR signal. The essential conditions and the effect produced are as follows:

	ESSENTIAL CONDITIONS		
Responding to:	Control Circuits Active	W field	Deactivate
ODR (simulated)	EF, Force EF	$\neq 0$ or $= 0$	EF, Force EF
ODR	Out Act	$1 \rightarrow 0$	Out Act
IDR	In Act	$1 \rightarrow 0$	In Act
ODR (quarter word)	Out Act C=01_2 or 11_2	$2 \rightarrow 1$	Out Act

Monitor interrupts occur when the I/O section detects certain control circuit conditions. The conditions detected and the effects produced upon their detection are as follows:

ESSENTIAL CONDITIONS		EFFECTS	
Active	Not Active	Interrupt Occurs	Deactivates
In Mon	In Act	ESI Input Monitor Interrupt	In Mon
Out Mon	Out Act	ESI Output Monitor Interrupt	Out Mon

7.3.10. Summary of the CPU's Operation Versus the EI, IDR, and ODR Control Signals

The EI control signal is turned on by the ESI subsystem when it is transmitting a status word to the CPU. The IDR control signal is turned on by the ESI subsystem when it is transmitting an input word to the CPU. The ESI subsystem turns on the ODR signal when it is ready to receive an output word from the CPU.

Note that a subsystem operating in ESI mode does not turn on an ODR signal when it is ready to receive a function word as was the case with a subsystem operating in ISI mode (see 7.3.5.1). The details of the CPU's recognition and response to ODR, IDR, and EI control signals are the same for ESI mode as for ISI mode and are explained in 7.2.10.

7.3.11. I/O Programming Considerations – ESI Mode

A function word transfer operation is performed via an output channel operating in ESI mode by execution of the LFC or LFCM instruction. To condition an output channel to perform an output word transfer operation, either the LOC or LOCM instruction is executed. To condition an input channel to perform an input word transfer operation, either the LIC or LICM instruction is executed.

Other instructions may also be executed in combination with the instructions stated in the preceding paragraph which determine the presence or absence of certain conditions on the specified channel. These instructions provide a program with the means to directly test the condition of a specified channel. These instructions are as follows:

- Jump Function Channel (JFC)

  When a JFC instruction is executed after an LFC or LFCM instruction, the JFC instruction tests the condition of the Force EF control circuit. If the Force EF control circuit is active, the program jumps to the location specified by the JFC instruction. If the Force EF control circuit is not active the program continues with the next consecutive instruction.

  In the operation of the CPU's I/O section, the Force EF control circuit is activated when the LFC or LFCM instruction is executed; it is deactivated during the I/O timing chain cycle which controls the transmission of the function word to the ESI subsystem following the execution of the LFC or LFCM instruction. In ESI mode, only one function word is transmitted via an ESI channel to the subsystem for each execution of the LFC or LFCM instruction.

- Jump Output Channel Busy (JOC)

  The JOC instruction determines whether or not an output channel operating in ESI mode is active in output mode. Specifically, the JOC instruction tests the condition of the Output Active control circuit. If the Output Active control circuit is active, the program jumps to the location U specified by the instruction. If the Output Active control circuit is not active, the program continues with the next consecutive instruction.

  In the operation of the CPU's I/O section, the Out Act control circuit for a channel operating in ESI mode is activated when the LOC or LOCM instruction is executed; it is deactivated when the DOC instruction is executed or when the terminal condition is detected in an associated ESI ACW. The Out Act control circuit is not activated by an LFC or LFCM instruction for a channel operating in ESI mode, as it was for a channel operating in ISI mode.

- Jump Input Channel Busy (JIC)

  Execution of the JIC instruction determines whether or not an input channel operating in ESI mode is active in input mode. Specifically, the JIC instruction tests the condition of the In Act control circuit. If the Input Active control circuit is active, the program jumps to the location U specified by the JIC instruction. If the Input Active control circuit is not active, the program continues with the next consecutive instruction.

  In the operation of the CPU's I/O section the Input Active Control circuit for a channel operating in ESI mode is activated when an LIC or LICM instruction is executed; it is deactivated when a DIC instruction is executed or when the terminal condition is detected in an associated ESI ACW.

7.3.11.1. ESI Function Word Transfer Programming Considerations

When a function word transmission operation is performed via an ESI channel, one function word is transmitted to the ESI subsystem via the specified channel each time the LFC or LFCM instruction is executed provided the W field of the associated OACW does not contain the value 0 (see 7.3.5.1).

The execution of the LFC or LFCM instructions neither activates nor deactivates the Output Active or Output Monitor control circuits on that channel. Therefore, the output mode as well as the input mode operation of that channel is unaffected.

Execution of the JFC instruction following execution of the LFC or LFCM instruction can be used to assure the program that the function word has been transmitted to the ESI subsystem. (In practice, a JFC instruction may appear in the coding just before the LFC or LFCM instruction. This assures that the channel can accept a function word before the LFC or LFCM is executed.)

7.3.11.2. ESI Output Word Transfer Programming Considerations

An output channel operating in ESI mode can be activated for an output word transmission by execution of either the LOC or LOCM instruction. The LOCM instruction is usually used to activate an output channel operating in ESI mode; it activates both the output mode and the output monitor function on the specified channel. When an output channel is operating with the output monitor function active, an ESI Output Monitor Interrupt occurs when the terminal condition is detected in an ACW associated with one of the devices operating via that channel.

When an ESI Output Monitor Interrupt occurs, the program can capture the number of the channel associated with that interrupt by executing the Store Channel Number (SCN) instruction. After the program has obtained the channel number, it also captures the address of the ESI ACW that has reached the terminal condition. This is possible because, when the ESI Output Monitor Interrupt occurs, the biased ESI address of the terminated ESI ACW is in bits 17 through 0 of the IACR associated with that channel.

Since the program can obtain the channel number and the address of the ESI ACW that has reached the terminal condition, it can, if so specified, reactivate that channel in output mode by execution of the LOC or LOCM instruction within a short enough time period so that there will be no delay in servicing ODR's presented by the ESI subsystem for any of the ESI devices on that channel. The program could also load a new ACW in the location of the ESI ACW that has reached the terminal condition and/or transmit a function word via the subsystem to the device associated by the ESI ACW. This action could cause the ESI device to change its mode of operation. The program could also choose to ignore the terminated ESI ACW; if so, the CPU's I/O section transmits an EOT word to the associated ESI device when that device presents an ODR and ESI word to the I/O section and the W field of the associated ESI ACW contains the value 0.

7.3.11.3. ESI Input Word Transfer Programming Considerations

An input channel operating in ESI mode can be activated for an input word transmission by execution of either the LIC or LICM instruction. The LICM instruction is usually used to activate an input channel operating in ESI mode; it activates both the input mode and the input monitor function on the specified channel. When an input channel is operating with the input monitor function active, an ESI Input Monitor Interrupt occurs when the terminal condition is detected in any one of the ESI ACW's.

When an ESI Input Monitor Interrupt occurs, the program can capture the number of the channel associated with that interrupt by executing the SCN instruction. After the program has obtained the channel number, it can also capture the address of the ESI ACW that has reached the terminal condition. This is possible because, when the ESI Input Monitor Interrupt occurs, the biased ESI word address of the terminated ESI ACW is in bit positions 35 through 18 of the IACR associated with that channel.

Since the program can obtain the channel number and the address of the ESI ACW that has reached the terminal condition, it can, if so specified, reactivate that channel in input mode by execution of the LIC or LICM instruction within a short enough time period so that there will be no delay in servicing IDR's presented by the ESI subsystem for any of the ESI devices on that channel. The program could also load a new ACW in the location of the ESI ACW that has reached the terminal condition and/or transmit a function word via the ESI subsystem to the associated ESI device which may turn off the device or change its mode of operation. If the program chooses to ignore the terminated ESI ACW, and if the ESI device, via the ESI subsystem continues to present IDR, ESI word, and input word signals to the CPU I/O section, the I/O section performs all the actions it normally performs when servicing an ESI IDR except that the associated ESI ACW is not updated and the input word is not transferred into main storage. However, allowing this idle activity to continue for a period of time is wasteful because the same amount of main storage cycle time is required to perform the idle activity as is required to actually transfer an input word into main storage.

7.3.12. ESI Input/Output Timing

When an ESI input or output transmission occurs, three main storage cycles are needed to perform the operation as follows:

- One main storage cycle is necessary to obtain the ESI ACW from the main storage location addressed by the biased ESI address.
- One main storage cycle is necessary to obtain an output word from, or to store an input word into, the location in main storage addressed by the ESI ACW.
- One main storage cycle is necessary to restore the updated ESI ACW into the main storage location addressed by the biased ESI address.

An elapsed time of three main storage cycles is required except for the following special case. If the ESI ACW is located in module pair #0 of main storage and the main storage location of the output word or input word is in one of the other module pairs, an elapsed time of two main storage cycles is needed. In this case only, the restoring of the ESI ACW is overlapped with the reading of an output word or the writing of an input word.

8. INTERRUPTS

8.1. INTRODUCTION

Interrupts are signals which queue the CPU control section when error conditions are detected by the hardware or when certain specified operations have been normally completed. The actions that are always performed by the CPU when it responds to an interrupt signal are as follows:

- All I/O interrupts are disabled.
- The current contents of the Processor State Register (PSR) are transferred to the addressable control register at address 000_8.
- D7 and D6 of the PSR are set to 1's; BI, BD, and BS remain unchanged; D8, D5 through D0, and QW are set to 0's.
- The normal sequence of instructions is interrupted and the CPU executes the instruction located in the main storage address assigned to the specific interrupt (see Table 3–3).

Normally, the instruction stored in the main storage location assigned to a specific interrupt will be either a Store Location and Jump (SLJ) or a Load Modifier and Jump (LMJ). Either of these two instructions can be used to capture the current contents of the P register and also perform a transition (jump) to some type of service routine. This routine can provide whatever service is required to handle the particular interrupt. When and if the particular circumstances call for returning control of the CPU to the interrupted program, the address captured by the SLJ or LMJ instruction can be used to determine the proper address at which to resume the interrupted program.

Interrupts are classed according to the events or conditions that triggered the interrupt. There are three general classes of interrupts:

- Input/Output interrupts (I/O interrupts)
- Fault interrupts
- Programmed interrupts

The I/O interrupts differ from the fault interrupts and the programmed interrupts in that they can be locked out either by the occurrence of a previous interrupt or by the execution of the Prevent All I/O Interrupts and Jump instruction. Furthermore, if an I/O interrupt occurs during the execution of an extended sequence such as a Block Transfer, the repeated search instructions or an Execute instruction, the interrupted instruction is terminated in an orderly manner so that it can be resumed when the interrupt condition has been satisfied.

By contrast, the fault interrupts and programmed interrupts occur without fail when the particular condition arises or when the Programmed Interrupt instruction is executed.

8.2. I/O INTERRUPTS

An I/O interrupt originates within the CPU or in a subsystem and they are presented to the CPU control section through the CPU interrupt priority network. Each interrupt informs the CPU of an abnormal condition or the normal completion of an assigned task. An I/O interrupt can interrupt the CPU sequence of instruction execution only when I/O interrupts are enabled. All I/O interrupts are automatically disabled when any interrupt occurs; they are also disabled when the Prevent All I/O Interrupts And Jump (PAIJ) instruction is executed. All I/O interrupts are enabled only by execution of the Allow All I/O Interrupts And Jump (AAIJ) instruction.

The four classes of I/O interrupts and the particular interrupts are:

- Monitor Interrupts
  - (1) ISI Input Monitor Interrupt
  - (2) ISI Output Monitor Interrupt
  - (3) ISI Function Monitor Interrupt
  - (4) ESI Input Monitor Interrupt
  - (5) ESI Output Monitor Interrupt
- External Interrupts
  - (1) ISI Channel External Interrupt
  - (2) ESI Channel External Interrupt
- System I/O Interrupts
  - (1) Interprocessor Interrupt
  - (2) RTC (Real Time Clock) Interrupt
  - (3) Dayclock Interrupt
  - (4) Power Loss Interrupt
- I/O Parity Error Interrupts
  - (1) ISI Access Control Word Parity Error Interrupt
  - (2) ESI Access Control Word Parity Error Interrupt
  - (3) I/O Data Parity Error Interrupt

8.2.1. Monitor Interrupts

The LICM instruction and the LOCM instruction are used to activate both the channel specified by the instruction and the monitor mode for that channel. For an ISI channel, the LFCM instruction is also used to activate the specified output channel and the output monitor mode for that channel. However, when a channel is operating in ESI mode and the LFCM is executed, the monitor mode is not activated for that channel. When an I/O transfer operation is performed with the monitor mode active, a monitor interrupt indicates that all the words initially specified by the contents of the W field of the associated ACW have been transferred.

The CPU responds to a monitor interrupt by performing the actions common to all interrupts, as described in 8.1. In addition, it stores the associated channel number in a nonaddressable register. This number can be obtained and stored in an addressable location by execution of the SCN instruction.

8.2.1.1. ISI Input Monitor Interrupt

An ISI input channel is deactivated, and the ISI Input Monitor Interrupt is triggered when both of the following conditions prevail:

(1) The currently active input transfer operation was initiated on the ISI channel by the execution of an LICM instruction.

(2) The subsystem presents an IDR to the CPU and, during the resulting input data word transfer operation, the contents of the associated ISI IACW is decremented from 1 to 0.

If the CPU's I/O section detects that the W field of the ISI IACW is zero at the time the input channel is activated and the CPU responds to an IDR signal presented on that input channel, then the input channel is deactivated, the ISI Input Monitor Interrupt is triggered, and the input word is not transferred into main storage.

The CPU I/O section responds to an ISI Input Monitor Interrupt by performing the instruction at MSR + 220_8 as its next instruction.

8.2.1.2. ISI Output Monitor Interrupt

An ISI Output channel is deactivated, and the ISI Output Monitor Interrupt is triggered when both of the following conditions prevail:

(1) The currently active output transfer was initiated on the ISI channel by the execution of an LOCM instruction.

(2) The CPU's I/O section responds to an ODR signal being presented on the ISI output channel and the contents of the W field of the associated ISI OACW is 0.

An output word is not transmitted to the subsystem in response to the ODR signal which triggered the ISI Output Monitor Interrupt.

The CPU's I/O section responds to an ISI Output Monitor Interrupt by performing the instruction at MSR + 221_8 as its next instruction.

8.2.1.3. ISI Function Monitor Interrupt

An ISI output chennel is deactivated, and the ISI Function Monitor Interrupt is triggered when both of the following conditions prevail:

(1) The currently active ISI output channel was activated as a result of executing the LFCM instruction.

(2) The CPU's I/O section responds to an ODR signal being presented on the ISI output channel and the contents of the W field of the associated ISI OACW is 0.

If the W field of the ISI OACW is zero when that channel is activated by an LFCM instruction, then the CPU's I/O section responds to the simulated ODR signal by deactivating that output channel and triggering the ISI Function Monitor Interrupt (see 7.2.6).

The CPU's I/O section responds to an ISI Function Monitor Interrupt by performing the instruction at MSR + 222_8 as its next instruction.

8.2.1.4. ESI Input Monitor Interrupt

An ESI input channel is deactivated, and the ESI Input Monitor interrupt is triggered when both of the following conditions prevail:

(1) The ESI input channel was activated as a result of executing the LICM instruction.

(2) The ESI subsystem presents an IDR to the CPU, and, during the resulting input word transfer operation, the contents of the W field of the associated ESI ACW is decremented from 1 to 0.

If the W field of the ESI ACW is zero when the associated ESI input channel is activated, then the ESI ACW is not updated, the input channel is not deactivated, an ESI monitor interrupt does not occur, and the input word is not transferred to main storage (see 7.3.6).

The CPU I/O section responds to an ESI Input Monitor Interrupt by performing the instruction at MSR + 224_8 as its next instruction.

8.2.1.5. ESI Output Monitor Interrupt

Normally an ESI output channel operating in half word or quarter word ESI mode is deactivated, and the ESI Output Monitor Interrupt is triggered when both of the following conditions prevail:

(1) The ESI output channel was activated as a result of executing the LOCM instruction.

(2) The ESI subsystem presents an ODR and ESI word signals to the CPU's I/O section, and, during the resulting output transfer operation, the contents of the W field of the associated ESI ACW is decremented from 1 to 0.

If the W field of the ESI ACW is zero when the associated ESI output channel is activated, then a half word (18 bits) of all 1 bits is sent via the function word/output word lines (17 through 0) of the associated output channel and an OA signal is sent via the OA line of that channel to the ESI subsystem. The ESI ACW is not updated, the output channel is not deactivated and an ESI Output Monitor interrupt does not occur (see 7.3.6).

However, there is an exception to the general case for quarter word ESI mode. If the C field of the ESI ACW contains either the value 01_2 or 11_2, the ESI channel will be deactivated and the ESI monitor interrupt will be triggered if the following conditions prevail:

(1) The ESI output channel was activated as a result of executing the LOCM instruction.

(2) The ESI subsystem presents an ODR signal and ESI word signals to the CPU's I/O section, and, during the resulting output word transfer, the contents of the W field of the associated quarter word ESI ACW is decremented from 2 to 1 (see 7.3.3.2).

The CPU I/O section responds to an ESI Output Monitor Interrupt by performing the instruction MSR + 225_8 as its next instruction.

8.2.2. External Interrupts

An external interrupt signal originates in a subsystem and is presented to the CPU via the priority network of the CPU's I/O section. It informs the CPU that a status word is on the input word/status word lines. The status word in turn informs the CPU of an error condition, the completion of a specific operation, or the existence of some other specific condition.

Some examples of situations in which an external interrupt is usually presented to the CPU are as follows:

- A subsystem received a function word from the CPU and the I/O operation could not be initiated in the subsystem.
- An I/O operation was initiated in a subsystem but could not be completed because one or more particular deterrent conditions were detected in the subsystem.
- An I/O operation was completed in a subsystem but an error condition occurred while the operation was being performed.
- An I/O operation was initiated by a function word which required the subsystem to activate the external interrupt signal upon completion of the specified I/O operation. (Many subsystems always present an external interrupt signal to the CPU's I/O section upon completion of an operation, regardless of whether or not the operation is completed satisfactorily.)

The CPU responds to an external interrupt by performing the activities common to all interrupts; i.e., disabling I/O interrupts and handling the PSR (see 8.1).

In addition, it stores the associated channel number in a nonaddressable register. This number can be obtained and stored in an addressable location by execution of the SCN instruction.

8.2.2.1. ISI External Interrupt

An ISI External Interrupt is triggered when the CPU's I/O section responds to an EI signal presented by a subsystem to the CPU's I/O section via the EI line of an ISI channel. When the ISI External Interrupt is triggered, a status word is automatically transferred via the input word/status word lines of that channel into a specific main storage location, and the CPU I/O section responds to the ISI External Interrupt by performing the instruction at MSR + 223_8 as its next instruction.

The EI signal which comes from the display console (always connected to I/O channel #15) triggers an ISI External Interrupt having the same basic characteristics as an ISI External Interrupt initiated by an EI signal from any other ISI I/O channel.

The status word conveys coded information to the CPU. For the Type 3011-95* and Type 3011-97 CPU's, the address at which the status word is stored depends on the number of the CPU being interrupted as specified by the PROC NO switches on the top row of the CPU maintenance panel. The address/CPU number relationships are as follows:

- Status Word stored at MSR + 200_8 for CPU #0
- Status Word stored at MSR + 201_8 for CPU #1
- Status Word stored at MSR + 202_8 for CPU #2

When the INTERRUPT ENABLE button on the Type 4004 operator's control console is depressed, the console is conditioned so that the external interrupt signal is turned on as soon as a character key on the console keyboard is depressed. Depressing a character key makes the character code available to the CPU via the input word/status word lines for storage in the status word main storage location associated with the CPU being interrupted.

8.2.2.2. ESI External Interrupt

An ESI External Interrupt is triggered when the CPU's I/O section responds to an EI signal presented by a subsystem to the CPU's I/O section, via the EI line of an ESI channel. When the ESI External Interrupt is triggered, a status word is automatically transferred from the input word/status word lines of that channel into the same main storage location as for an ISI External Interrupt (see 8.2.2.1), and the CPU responds to the ESI External Interrupt by performing the instruction as MSR + 227_8 as its next instruction.

8.2.3. System I/O Interrupts

The Interprocessor Interrupt, Real Time Clock (RTC) Interrupt, Day Clock Interrupt and Power Loss Interrupt are called system I/O interrupts because they can occur only when I/O interrupts are enabled, even though they are not related to any specific I/O channel or I/O activity.

The actions automatically performed by the CPU when it responds to any system I/O interrupt signal are described in 8.1.

**For the Type 3011–99 CPU, the status word is always stored at MSR + 230.*

8.2.3.1. Interprocessor Interrupt

An Interprocessor Interrupt occurs following receipt of an Interprocessor Interrupt signal by one CPU in a multiprocessor system from one of the other CPU's in the system. The Interprocessor Interrupt signal is turned on when an Initiate Interprocessor Interrupt (III) instruction is performed. The CPU receiving the Interprocessor Interrupt signal responds by acknowledging it and performing the instruction at either MSR + 232_8 or MSR + 233_8 (depending on the CPU #'s involved and the value in the a field of the III instruction – see 6.14.5) as its next instruction.

8.2.3.2. Real Time Clock Interrupt

The Real Time Clock (RTC) is a timing device within the CPU. The RTC can be enabled or disabled manually by operation of the RTC switch at the display console. When the CPU is operating in the real time mode as specified at the Availability Control Unit or the CPU maintenance panel, it is impossible to disable the RTC.

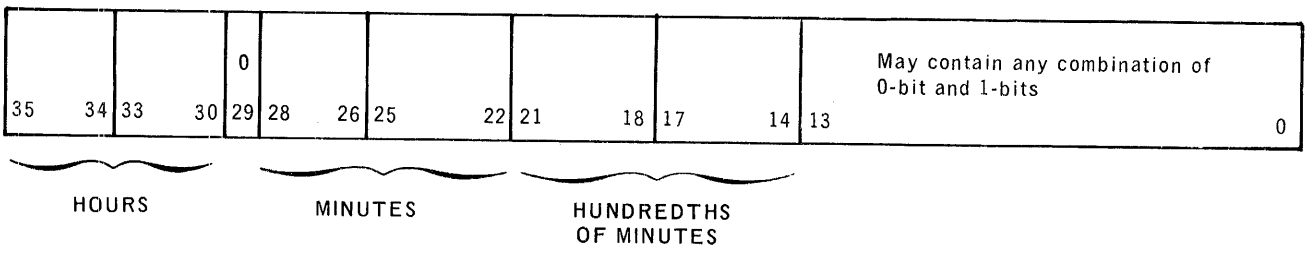
When the RTC is enabled, the content of bits 17 through 0 of the RTC Register (the control register at address 100_8) is decremented by 1 every 200 microseconds. The RTC Interrupt is triggered when the CPU detects that the contents of bits 17 through 0 of the RTC Register are decremented from 000000_8 to 777776_8. The CPU's I/O section responds to an RTC Interrupt by executing the instruction at MSR + 231_8 as the next instruction. (The contents of bits 35 through 18 of the RTC Register are not affected by RTC decrementation.)

8.2.3.3. Day Clock Interrupt

The Day Clock Interrupt signal is controlled by the day clock associated with the operator's display console and is presented to the CPU's I/O section via a special signal in the I/O cabling for channel #15.

All inputs from the day clock can be disabled by performing a Disable Day Clock instruction. Performing an Enable Day Clock instruction enables inputs from the day clock provided it is running (it has not been stopped by means of the DAY CLOCK STOP switch on the display console) and inputs have not been disabled by means of the DISABLE DAY CLOCK switch on the CPU maintenance panel.

When the day clock is running, it is updated each 600 ms (.01 minutes). If it is not disabled, the updated time is transferred to main storage location MSR + 216_8 in the format shown below. The update and input interval can be changed from 600 ms to 6 seconds or one minute by a wiring change in the display console. The interrupt interval can also be changed by a wiring change as explained in the following paragraph.



When the day clock is running and has not been disabled, the Day Clock Interrupt is turned on at 6-second intervals (when the low order hundredth of minutes digit produces a carry to the next higher time digit). The CPU responds to the Day Clock Interrupt by acknowledging it and executing the instruction at main storage location MSR + 217_8 as its next instruction. The interrupt interval can be changed from six seconds to 600 milliseconds, 10 minutes, one hour, or 24 hours by a wiring change in the display console.

8.2.3.4. Power Loss Interrupt

The Power Loss Interrupt is turned on when power sensing hardware in the CPU detects that the CPU voltages are falling below a prescribed level. The CPU responds by executing the next instruction from the main storage location at address MSR + 210_8. When the Power Loss Interrupt signal is turned on, approximately five milliseconds of program operating time remain to prepare for the predicted loss of all electrical power. Since the contents of the addressable control registers are destroyed and the contents of main storage remains unchanged when power is lost, it is possible to use the five milliseconds for a programmed routine which transfers the contents of the control registers and other pertinent data to main storage. During these five milliseconds, the program may store enough information so that when power is restored to the CPU, it is possible to resume many programs.

8.2.4. I/O Parity Error Interrupts

There are three I/O parity error interrupts: the ISI Access Control Word Parity Error Interrupt, the ESI Access Control Word Parity Error Interrupt, and the I/O Data Parity Error Interrupt. The CPU responds to an I/O parity error interrupt by performing the activities common to all interrupts; that is, disabling the I/O interrupts and handling the PSR (see 8.1). In addition, it stores the associated channel number in a nonaddressable register. This number can be obtained and stored in an addressable location by execution of the SCN instruction.

8.2.4.1. ESI Access Control Word Parity Error Interrupt

An ESI Access Control Word Parity Error Interrupt occurs following detection of a parity error in the ESI ACW read from main storage during an ESI mode input word or output word transfer operation. In addition to the activities involving the disabling of I/O interrupts, the handling of the PSR, and the storing of the channel number associated with the I/O parity error, the CPU does the following:

- If the parity error is detected in an ESI input ACW read from main storage during an input transfer sequence,
  - (1) an IA signal is sent to the subsystem;
  - (2) the input word is not written into main storage;
  - (3) the ESI ACW in main storage remains unchanged;
  - (4) the associated input channel is not deactivated; the ESI Input Monitor Interrupt does not occur.

- If the parity error is detected in an ESI output ACW read from main storage during an output transfer sequence,

  (1) an OA signal is sent to the subsystem;

  (2) a word of all 1's is sent to the ESI subsystem via the output word/function word lines rather than the word addressed by the ESI ACW;

  (3) the ESI ACW in main storage remains unchanged;

  (4) the associated output channel is not deactivated; the ESI Output Monitor Interrupt does not occur.

  The CPU I/O section responds to an ESI Access Control Word Parity Error Interrupt by executing the instruction at MSR + 211_8 as its next instruction.

8.2.4.2. ISI Access Control Word Parity Error Interrupt

An ISI Access Control Word Parity Error Interrupt occurs following detection of a parity error in the ACW read from an IACR or OACR during an ISI input or output transfer sequence. The ISI Access Control Word Parity Error Interrupt can occur during an ISI input word, ISI output word, ISI function word, or ESI function word transfer sequence. In addition to the activities involving the disabling of I/O interrupts, the handling of the PSR (as described in 8.1), and the storing of the channel number associated with the I/O parity error, the CPU does the following:

- If the parity error is detected in an ISI IACW read from an IACR during an input transfer sequence,

  (1) an IA signal is sent to the subsystem;

  (2) the input word is not written into main storage;

  (3) the contents of the IACR remains unchanged;

  (4) the associated input channel is deactivated; the ISI Input Monitor Interrupt does not occur.

- If the parity error is detected in an ISI OACW read from an OACR during an output data transfer sequence,

  (1) an OA signal is sent to the subsystem;

  (2) a word of all 1's is sent to the subsystem via the output word/function word lines rather than the word addressed by the OACW;

  (3) the contents of the OACR remain unchanged;

  (4) the associated output channel is deactivated; the ISI Output Monitor Interrupt does not occur.

- If the parity error is detected in an OACW read from an OACR during an ISI function word or ESI function word transfer sequence,

  (1) the EF signal is not sent to the subsystem;

  (2) a function word is not sent to the subsystem;

  (3) the contents of the OACR remain unchanged;

  (4) for the aborted ISI function word transfer sequence, the associated output channel is deactivated; the ISI Function Monitor Interrupt does not occur. For the aborted ESI function word transfer sequence, the associated output channel is not deactivated.

The CPU's I/O section responds to an ISI Access Control Word Parity Error Interrupt by executing the instruction at MSR + 212_8 as its next instruction.

8.2.4.3. I/O Data Parity Error Interrupt

An I/O Data Parity Error Interrupt occurs when a parity error is detected in an ISI or ESI output data word or function word read from main storage during an output transfer sequence. An I/O Data Parity Error Interrupt also occurs if a parity error is detected during the read portion of the main storage read/write cycle when an ESI input word transmission operation is being performed. (These are always partial word operations.) In addition to the activities involving the disabling of I/O interrupts, the handling of the PSR (as described in 8.1), and the storing of the channel number associated with the I/O parity error, the CPU does the following:

- If the parity error is detected during an ISI output data word transfer sequence,

  (1) an OA signal is sent to the subsystem;

  (2) the data bits of the output data word containing the error are sent to the subsystem;

  (3) the contents of the associated OACR are updated;

  (4) the associated output channel is normally not deactivated; the ISI Output Monitor Interrupt normally does not occur. (However, the associated output channel is deactivated and the corresponding output monitor can occur if the terminal condition is detected in the updated ACW.)

- If the parity error is detected in an ESI output data word transfer sequence,

  (1) an OA signal is sent to the subsystem;

  (2) the specified data bits of the output data word containing the error are sent to the subsystem;

  (3) the contents of the associated ESI ACW in main storage are updated;

  (4) the associated output channel is normally not deactivated; the ESI Output Monitor Interrupt normally does not occur. (However, the associated output channel is deactivated and the corresponding output monitor can occur if the terminal condition is detected in the updated ACW.)

- If the parity error is detected during an ISI or ESI function word transfer sequence,

  (1) an EF signal is not sent to the subsystem;

  (2) the function word is not sent to the subsystem;

  (3) the contents of the related OACR are updated;

  (4) the associated output channel is not deactivated; the Function Monitor Interrupt does not occur.

- If the parity error is detected in the word read during the read portion of the read/write data store cycle for an ESI input transfer operation, the following occurs:

  (1) an IA signal is sent to the subsystem;

  (2) the writing of the input word into main storage is completed;

  (3) the contents of the ESI ACW in main storage is updated;

  (4) the associated input channel normally is not deactivated; the ESI Input Monitor Interrupt normally does not occur. However, the associated input channel is deactivated and the ESI Input Monitor Interrupt can occur if the terminal condition is detected in the updated ACW.

NOTE: For a half word ESI input operation, the I/O Data Parity Error Interrupt occurs only if the parity error is detected in that half of the main storage location not currently being addressed by the H and V field of the ESI half word ACW. For a quarter word ESI input operation, the I/O Parity Error Interrupt occurs if the parity error is detected in either half of the main storage location currently being addressed by the V field of the ESI quarter word ACW.

The CPU I/O section responds to an I/O Data Parity Error Interrupt by executing the instruction at MSR + 213_8 as its next instruction.

8.2.5. I/O Interrupt Priority

All I/O interrupts, IDR's, and ODR's are subject to the operation of the I/O interrupt priority network of the CPU's I/O section. The ODR's and IDR's command a higher priority than the I/O interrupts. The priority from highest to lowest is the order listed in Table 8–1 except that an IDR may have higher priority than an ODR under certain circumstances, as explained in 7.1.7.

PRIORITY	NAME OF REQUEST OR INTERRUPT
1	Output Data Request (ODR)
2	Input Data Request (IDR)
3	Real Time Clock Decrement
4	Power Loss
5	I/O Parity Error Interrupt
6	ESI External Interrupt
7	ESI Input Monitor Interrupt
8	ESI Output Monitor Interrupt
9	Real Time Clock Interrupt
10	ISI External Interrupt
11	ISI Input Monitor Interrupt
12	ISI Output Monitor Interrupt
13	ISI Function Monitor Interrupt
14	Interprocessor Interrupt #0
15	Interprocessor Interrupt #1

Table 8–1. I/O Request and Interrupt Priority Table

If two or more interrupt signals of the same priority level are simultaneously active on different channels, priority is determined on the basis of channel number and the interrupt associated with the numerically lower numbered channel has the higher priority.

At the moment that the CPU responds to any interrupt, all I/O interrupts are disabled. The I/O interrupt signals that may be turned on while the I/O interrupts are disabled must wait until all I/O interrupts are enabled again, at which time the waiting I/O interrupt signal with highest priority is responded to by the I/O section. No interrupt signals are lost, but each is responded to when its priority is higher than the priority of any other waiting interrupt.

NOTE: If an I/O parity error is detected while the CPU is still servicing a previously detected I/O parity error, and if all I/O interrupts are still disabled, the channel number associated with the most current I/O parity error is not available. The instruction from the main storage interrupt location for the most current parity error is not executed.

8.3. FAULT INTERRUPTS

Fault interrupt signals are generated within a CPU when specific conditions exist which would cause the results obtained to be erroneous if the current program is continued. Fault interrupts are never disabled, but if more than one fault interrupt generating condition is detected during the execution of an instruction, the CPU responds to the first one detected, except as noted under the priorities for main storage and control register parity error interrupts, Table 8–2.

Fault interrupts are divided into two groups;

- main storage and control register parity error interrupts
- program error fault interrupts

NONINTERLEAVED MAIN STORAGE		INTERLEAVED MAIN STORAGE		INTERRUPT LOCATION	PRIORITY IN CASE OF MULTIPLE ERRORS DURING AN INSTRUCTION
MAIN STORAGE LOGICAL MODULE #	ADDRESS RANGE	MAIN STORAGE LOGICAL MODULE #	ADDRESS RANGE		
0	0-077,777$_8$	0	0-177,777	(LAR)* + 077,776$_8$	2
1	100,000-177,777	2	400,000-577,777	MSR + 235$_8$	3
2	200,000-277,777	1	200,000-377,777	MSR + 236$_8$	4
3	300,000-377,777	3	500,000-777,777	MSR + 237$_8$	5
CONTROL REGISTER				MSR + 240$_8$	1

NOTE: There are four LAST ADDRESS toggle switches on the CPU's maintenance panel. The leftmost three switches correspond to bits 17 through 15 of the 18-bit absolute address for this Main Storage Parity Error Interrupt. The rightmost switch corresponds to bit 14 of the interrupt address. (If a LAST ADDRESS switch is in the up position, it corresponds to a 1 bit; if it is in the down position, it corresponds to a 0 bit.) In Table 8–2, it is assumed that the LAST ADDRESS switches for bits 17 through 15 are in the down position and the LAST ADDRESS switch for bit 14 is in the up position. The CPU's hardware determines each bit in positions 17 through 15 of the interrupt address by forming the logical OR of the corresponding bit in the Last Address Register (LAR) and the setting of the corresponding LAST ADDRESS switch. Bit position 14 of the interrupt address is solely dependent on the setting of the LAST ADDRESS switch for bit 14.

Table 8–2. Main Storage or Control Register Parity Error Fault Interrupt Locations and Priorities

8.3.1. Main Storage and Control Register Parity Error Interrupts

Each main storage location and each addressable control register stores 38 bits consisting of 36 data bits and 2 parity bits. One of the parity bits is associated with the data bits in bits 35 through 18; the other parity bit is associated with the data bits in bits 17 through 0. When data is stored, the values for each of the two parity bits are generated based on the contents of the associated data bit positions.

When a data word or instruction word is read, the values for the two parity bits are generated based on the values read from bits 35 through 18 and 17 through 0, respectively. The generated parity bits are then compared with the parity bits read; if either or both of the generated parity bits are not equal to the parity bits read, a parity error interrupt signal is generated.

8.3.1.1. Main Storage Parity Error Interrupt

A Main Storage Parity Error Interrupt is generated in the event that a parity error is detected when program data (an instruction word or an operand or result word) is read from main storage.

These parity errors are distinguished from I/O data parity errors in that they do not involve transfers of data to or from an I/O device. The main storage parity errors occur when data is read from main storage to be used as an operand, as a result, or as an instruction word. A Main Storage Parity Error Interrupt is never generated when a full word write into a main storage location is performed, but it can be generated when a partial word write (1/2, 1/3, 1/4, or 1/6 word) into main storage is performed. The hardware performs a partial word write into main storage by:

(1) reading the 36-bit data word and the 2 parity bits from the specified main storage location;

(2) modifying the word read by replacing the contents of the specified bit positions with the specified partial word and replacing the contents of the associated parity bit with the proper value; and,

(3) storing the 36-bit modified word and the 2 parity bits back into main storage.

The test for parity error is performed on the data word read from main storage as described in step 1. For a half word write operation, the generation of a Main Storage Parity Error Interrupt indicates that a parity error has been detected in that half of the full word which is not to be changed by the partial write. For a third, quarter, or sixth word write operation, the generation of a Main Storage Parity Error Interrupt indicates that a parity error has been detected in either or both halves of the word read from a main storage location and a data error can exist in any bit position or positions of the contents of that full word read including the parity bit positions.

When the CPU reacts to a Main Storage Parity Error Interrupt, the actions performed by the CPU are those described in 8.1, and in the following:

- The system fault alarm sounds.
- The appropriate fault indicators on both the CPU's maintenance panel and the display console light.

The CPU responds to a Main Storage Parity Error Interrupt by executing the instruction from the main storage location related to the main storage module from which the word in error was read, as explained in 8.3.1.3.

8.3.1.2. Control Register Parity Error Interrupt

A Control Register Parity Error Interrupt is triggered in the event that a parity error is detected when a word is read from one of the 128 addressable control registers for an instruction or for a real time clock update. A Control Register Parity Error Interrupt is never generated when a write into an addressable control register is performed.

When the CPU responds to a Control Register Parity Error Interrupt, the actions performed are those described in 8.1 and the following:

- The system fault alarm sounds.
- The ICR fault indicators on the CPU maintenance panel and the display console light.

The CPU responds to a control register parity error by executing the instruction at MSR + 240_8 as its next instruction.

8.3.1.3. Instruction Locations and Priority

The addresses of the main storage locations from which the next instruction is obtained when a Main Storage or Control Register Parity Error Interrupt occurs are as shown in Table 8-2 Since the detection of a main storage or control register parity error does not disable other parity error detections, it is possible for more than one parity error to be detected during the execution of an instruction. When two or more parity errors are detected during the execution of an instruction, only one parity error interrupt signal is generated. In that case, the parity error to which the CPU responds is determined by the order of priority as listed in Table 8–2.

8.3.1.4. Considerations Related to a Main Storage Parity Error

- If a parity error is detected when reading an instruction from main storage, the instruction is not executed. The instruction with parity error is written back into memory during the write portion of the read/write cycle.
- If a main storage parity error is detected when reading an operand, the instruction being executed is completed including the transfer of the operand word regardless of whether the word had proper parity.
- When an instruction specifies a read (load), new parity is not generated for the write portion of the read/write cycle; therefore, any attempt to reread the contents of that location which produced the parity error should produce another parity error.
- When an instruction specifies a partial write (store), new parity is generated for only that half of the word that is altered. If a main storage parity error is detected during the read portion of the partial word write operation, the word read with incorrect parity will be restored with correct parity if the parity error was detected in that half of the word that is changed in the partial write operation. In this case, attempts to reread that word normally will not produce another main storage parity error.

8.3.1.5. Considerations Related to a Control Register Parity Error

- When a control register parity error is detected during the execution of one of the store instructions, the storing of the data into main storage is not performed.
- If a control register parity error is detected when reading data from a control register, a repeated read, or reads from that control register, may or may not result in repeated control register parity errors.
- In the execution of an instruction, a control register parity error may be detected in an index register (X register) referenced by either the x field or a field of the instruction, even if the x field or a field contains the value zero. For example, in the execution of an instruction such as Store X, both the x field and the a field are read and checked for parity. If the x field is zero, the PSR word temporary storage register at control register address 0000_8 is read and checked for parity. Thus, a control register parity error could occur on either the control register referenced by the x field or the control register referenced by the a field.
- Correct parity can be restored in a control register by writing data into that control register. Correct parity is always restored in all control registers by manually pressing the MASTER CLEAR switch on the display console or by manually pressing the ICR CLEAR switch on the CPU's maintenance panel. Either of these operations clear the control registers to all zero data bits with proper parity. If the CPU is operating in a multiprocessor configuration defined by the settings on an Availability Control Unit, the control registers of the CPU are master cleared each time the ACU initiates the automatic recovery procedure.

8.3.2. Program Error Fault Interrupts

Program error fault interrupts are as follows:

- Illegal Instruction Fault Interrupt
- Guard Mode Fault Interrupt
- Floating-Point Characteristic Underflow Fault Interrupt
- Floating-Point Characteristic Overflow Fault Interrupt
- Divide Fault Interrupt

8.3.2.1. Illegal Instruction Fault Interrupt

The Illegal Instruction Fault Interrupt is triggered when the CPU's control section detects an undefined function code in an instruction (see 6.15).

When an Illegal Instruction Fault Interrupt occurs, the CPU performs those actions common to all interrupts; that is, it disables all I/O interrupts, stores the current PSR and resets the PSR (see 8.1). In addition, it responds to the Illegal Instruction Fault Interrupt by executing the instruction at MSR + 241_8 as its next instruction.

- If the CPU is not operating in guard mode, the system fault alarm sounds and the system fault alarm indicators on the CPU's maintenance panel and display console light.
- If the CPU is operating in guard mode, these special alarms are not turned on.

8.3.2.2. Guard Mode/Storage Limits Protection Fault Interrupt

The Guard Mode/Storage Limits Protection Fault Interrupt occurs when the CPU is operating in guard mode (D2=1) and an attempt is made to perform one of the following instructions or actions.

(1) any I/O instruction (f = 75, all j-values)

(2) Prevent All I/O Interrupts And Jump

(3) Load Processor State Register

(4) Load Storage Limits Register

(5) Initiate Interprocessor Interrupt

(6) Alarm

(7) Disable Day Clock

(8) Enable Day Clock

(9) Select Interrupt Locations

(10) Load Channel Select Register

(11) Load Last Address Register

(12) instructions containing certain illegal codes (f,j = 75,13; 75,16; or 75,17 – see 6.15)

(13) a write into any control register location 40_8 through 100_8 and 120_8 through 177_8

(14) Disabling all I/O interrupts for more than 100 usec. by a looped or cascaded series of Execute instructions and/or indirect addressing sequences. Performing an Execute instruction or an indirect addressing sequence automatically locks out all I/O interrupts until the instruction has been completed.

The Guard Mode/Storage Limits Protection Fault Interrupt also occurs if specific types of main storage references which violate the Storage Limits Register are attempted when the PSR contains specific combinations of values for D3 and D2 (storage limits protection and guard mode), as follows:

(1) if a write operation is attempted in a protected area when D3D2 = 10 or 11

(2) if an operand read, a write, or a jump to operation is attempted in a protected area when D3D2 = 01

When a Guard Mode/Storage Limits Protection Fault Interrupt occurs, the CPU performs those actions common to all interrupts; i.e., it disables all I/O interrupts, stores the current PSR, and resets the PSR (see 8.1). In addition, it responds to the Guard Mode/Storage Limits Protection Fault Interrupt by executing the instruction at MSR + 243_8 as its next instruction.

8.3.2.3. Floating-Point Characteristic Underflow Fault Interrupt

The Floating-Point Characteristic Underflow Fault Interrupt is triggered only if the CPU detects that the characteristic of the result obtained is less than zero for either a single-precision floating-point or a double-precision floating-point operation. (See 4.4.4.1.1.)

When the CPU responds to a Floating-Point Characteristic Underflow Fault Interrupt, the CPU performs the actions described in 8.1. The instruction from the main storage location MSR + 245_8 is performed as the next instruction.

NOTE: If a floating-point characteristic underflow fault is detected for an instruction which specifies index register modification, the index modification is carried through to normal completion. When a floating-point characteristic underflow fault is detected, the result is not stored and the contents of the original (input) operand locations remain undisturbed.

8.3.2.4. Floating-Point Characteristic Overflow Fault Interrupt

The Floating-Point Characteristic Overflow Fault Interrupt is triggered during the execution of a floating-point instruction only if the CPU detects that the characteristic of the result obtained is greater than 377_8 for a single-precision floating-point operation or 3777_8 for a double-precision floating-point operation.

When the CPU responds to a Floating-Point Characteristic Overflow Fault Interrupt, the CPU performs the actions described in 8.1. The instruction from the main storage location MSR + 246_8 is performed as the next instruction.

NOTE: If a floating-point characteristic overflow fault is detected for an instruction which specifies index register modification, the index modification is carried through to normal completion.

8.3.2.5. Divide Fault Interrupt

The Divide Fault Interrupt is triggered when the CPU detects that more than 36 bits would be required to properly express the signed quotient for a fixed-point divide instruction or that division by zero is being attempted.

The Divide Fault Interrupt occurs in the following four specific cases:

- For the Divide Integer instruction, when the absolute value of the dividend is greater than or equal to 2^{35} times the absolute value of the divisor.
- For the Divide Single Fractional instruction, when the absolute value of the divisor is less than or equal to the absolute value of the dividend.

- For the Divide Fractional instruction, when the absolute value of the divisor is less than or equal to the absolute value of the most significant word of the dividend.

- For either floating-point division instruction, when the signed mantissa is plus or minus zero.

The first three cases listed include the cases in which the divisor for any fixed-point divide instruction is plus or minus zero.

When the CPU responds to a Divide Fault Interrupt, the actions performed by the CPU are those described in 8.1. The instruction from the main storage location at MSR + 247_8 is performed as the next instruction.

NOTE: Detection of a divide fault condition during execution of either floating-point divide instruction inhibits any reaction to a floating-point characteristic overflow or underflow fault condition. Only the Divide Fault Interrupt occurs.

8.4. PROGRAMMED INTERRUPTS

The programmed interrupts are as follows:

- Executive Return Interrupt

- Test and Set Interrupt

8.4.1. Executive Return Interrupt

The Executive Return Interrupt is generated when the Executive Return (ER) instruction is executed. The operation performed by the ER instruction provides the user program with the ability to transfer control to the executive program.

The CPU responds to the Executive Return Interrupt by performing the actions described in 8.1. The instruction from the main storage location MSR + 242_8 is performed as the next instruction.

8.4.2. Test and Set Interrupt

The Test and Set Interrupt is triggered when the Test and Set (TS) instruction is executed and bit 30 of the main storage location tested by the instruction contains a 1 bit. The operation performed by the TS instruction provides control over multi-processor use of common Data or common program segments.

The CPU responds to a Test and Set Interrupt by performing the actions described in 8.1. The instruction from the main storage location MSR + 244_8 is performed as the next instruction.

8.5. P REGISTER CONTENTS CAPTURED AT INTERRUPTS

The main storage locations associated with the various types of interrupts are listed in Table 3–3. Proper program response to any interrupt requires that the associated main storage location contain an LMJ or SLJ instruction to capture and store an address related to the point of interruption and initiate the response to the interrupt.

Table 8–3 lists various individual interrupts and types of interrupts and indicates the address which is captured by the LMJ or SLJ instruction contained in the main storage interrupt location for that particular interrupt. For some interrupts, the address captured depends on the particular instructions performed immediately preceding the LMJ or SLJ instruction. In these cases the address captured is shown for the various possible preceding instructions.

The address captured is always a relative address (see 9.3.5). It is represented in the table by one of the values P, P+1, P+2, or a jump to address. In this notation, P normally represents the address of the instruction which just precedes the interrupt. If the interrupt was caused by detection of some error or abnormal condition which prevented completion of the instruction, P normally represents the address of that instruction. An exception to these statements occurs when the interrupted instruction was loaded into the control section as a result of an Execute instruction. In this case, P represents the address of the Execute instruction. For those cases in which the address captured is shown as a jump to address, it is the address generated as the result of performing a jump instruction just before the interrupt and bears no direct relationship to the address of that instruction.

8.6. PROGRAM CONSIDERATIONS FOR HANDLING INTERRUPTS

The address captured and stored by the LMJ or SLJ instruction contained in the main storage location associated with any particular interrupt is a relative address derived from the contents of the P register as indicated in Table 8–3. (See 9.3.5.) The LMJ or SLJ instruction also loads the P register with an absolute jump to address. This absolute address is the address of the entrance to the routine used for the program response to the particular interrupt.

When an interrupt occurs, the processor state word for the program which has been interrupted is automatically stored and a processor state word suitable for an interrupt handling routine is formed in the PSR. This processor state word is used as required during each phase of the LMJ or SLJ instruction (and each subsequent instruction) until the contents of the PSR are changed. The following hardware characteristics are important considerations related to the interrupt handling routine.

- The values for D7 and D6 are set to 1's. The values for D8, D5 through D0, and QW in the PSR are set to 0's.
- The values for BI, BD, and BS are not affected when an interrupt occurs.
- When a Monitor Interrupt, an External Interrupt, or an I/O Parity Error Interrupt occurs, the CPU loads a nonaddressable register with the associated I/O channel number.
- All I/O interrupts are disabled when any interrupt occurs.

INTERRUPT NAME	P REGISTER CONTENTS CAPTURED			
	P	P+1	P+2	JUMP TO ADDRESS
All I/O Type Interrupts:				
Repeated Instructions In Progress	X			
In Termination Pass, Skip Not Set (Go to NI)		X		
In Termination Pass, Skip Set (Skip NI)			X	
Test Instructions, Go to NI		X		
Test Instructions, Skip NI			X	
Unconditional Jump Instruction				X
Conditional Jump Instruction, Go to NI		X		
Conditional Jump Instruction, Jump				X
All Other Instructions		X		
Main Storage Parity Error Fault:				
Instruction Word Parity Error:				
All Instructions		X		
Operand or Result Word Parity Error:				
Repeated Instruction in Progress		X		
In Termination Pass, Skip Not Set (Go to NI)		X		
In Termination Pass, Skip Set (Skip NI)		X		
Test Instruction, Skip Not Set (Go to NI)		X	X①	
Test Instruction, Skip Set (Skip NI)		X	X①	
All Other Instructions		X	X①	
Control Register Parity Error Fault:				
Repeated Instruction In Progress		X		
In Termination Pass, Skip Not Set (Go to NI)		X		
In Termination Pass, Skip Set (Skip NI)			X	
Test Instruction, Skip Not Set (Go to NI)		X		
Test Instruction, Skip Set (Skip NI)			X	
Unconditional Jump Instruction				X
Conditional Jump Instruction, Go to NI		X		
Conditional Jump Instruction, Jump				X
All Other Instructions		X	X①	
Illegal Instruction		X		
Guard Mode Fault:				
Repeated Instruction In Progress		X		
In Termination Pass, Skip Not Set (Go to NI)		X		
In Termination Pass, Skip Set (Skip NI)		X	X②	
Test Instruction, Skip Not Set (Go to NI)		X		
Test Instruction, Skip Set (Skip NI)		X	X②	
Unconditional Jump Instruction		X		
Conditional Jump Instruction, Go to NI		X		
Conditional Jump Instruction, Jump		X		
All Other Instructions		X	X①	
Characteristic Underflow			X	
Characteristic Overflow			X	
Divide Fault			X	
Executive Return		X		
Test and Set			X	

NOTES: ① The address captured is represented by P+2 when the operation of reading or writing the operand for the current instruction overlaps the operation of reading the next instruction word from main storage (alternate bank operation applies). It is represented by P+1 when same bank operation applies. In this context, current instruction is the instruction which lead to detection of the fault or parity error associated with the interrupt.

② In this case, the address captured can be either of the values represented by P+1 or P+2. The value captured depends on the exact nature of the fault which led to the interrupt and the exact moment of detection during the instruction. It does not depend on same bank versus alternate bank timing.

Table 8–3. UNIVAC 1108 Interrupt Versus P Register Contents Captured

D7 is set to 1 when an interrupt occurs. This action converts the interpretation of i=1 from indirect addressing to base register suppression. When base register suppression is effective (D7 = 1), $x \neq 0$, and i = 1, then each absolute jump to address, operand word address, or result word address formed in the index subsection is $u + X_m$ (and $u + X_m + 1$ for double-precision instructions). When base register suppression is effective (D7 = 1), x = 0, i = 1 and $j \neq 16$ or 17 for $f = 01 - 67_8$, then the address formed is u (and u + 1 for double-precision instructions). Thus the CPU can develop absolute addresses which are independent of the values for BI, BD, and BS in the processor state word which were carried forward from the interrupt program.

D6 is set to 1 when an interrupt occurs so that implicitly addressed R registers and the A, X, and R registers addressed by the a and x fields of the subsequent instructions are from the executive set of A, X, and R registers (addresses 120_8 through 175_8) rather than from the user set (addresses 001_8 through 035_8 and 100_8 through 117_8).

The PSR is changed when an interrupt occurs, so that, in most cases, it must be loaded with the values appropriate to the interrupted program before control is returned to that program.

During a Monitor Interrupt, an External Interrupt, or an I/O Parity Error Interrupt, the channel number associated with that interrupt is stored in a nonaddressable register. This number can be captured by the interrupt routine by means of the Store Channel Number (SCN) instruction. The SCN instruction must be executed while the I/O interrupts are disabled for that particular interrupt if the channel number is to be preserved. Otherwise, a subsequent interrupt could replace that channel number in the nonaddressable register.

Since all I/O interrupts are disabled when an interrupt occurs, the Allow All I/O Interrupts And Jump instruction should be performed as soon as the interrupt program is logically interruptible. Barring manual intervention or initiation of the auto recovery procedure by the Availability Control Unit, this instruction must be performed before an I/O interrupt can occur.

9. EXECUTIVE CONTROL

9.1. GENERAL

The CPU was designed to be run under control of an executive program which:

(a) assigns the absolute main storage locations to user programs and data;

(b) handles all I/O transfers.

As a result, certain control capabilities are provided for the exclusive use of the Executive program. The hardware for addressing utilizes base registers which allow the Executive program to relocate any program and/or its data in main storage and to subsequently run that program without modification. These base registers are contained in the Processor State Register (PSR). This section defines and explains the various operations related to and affected by the contents of the PSR and other registers and operations under Executive control.

9.2. PROCESSOR STATE REGISTER

The PSR is a 36-bit register which contains the two base registers BI and BD, used in base indexing and also some special designators which define various states and conditions affecting the current operation of the CPU. The format of the processor state word is given in Figure 9–1.

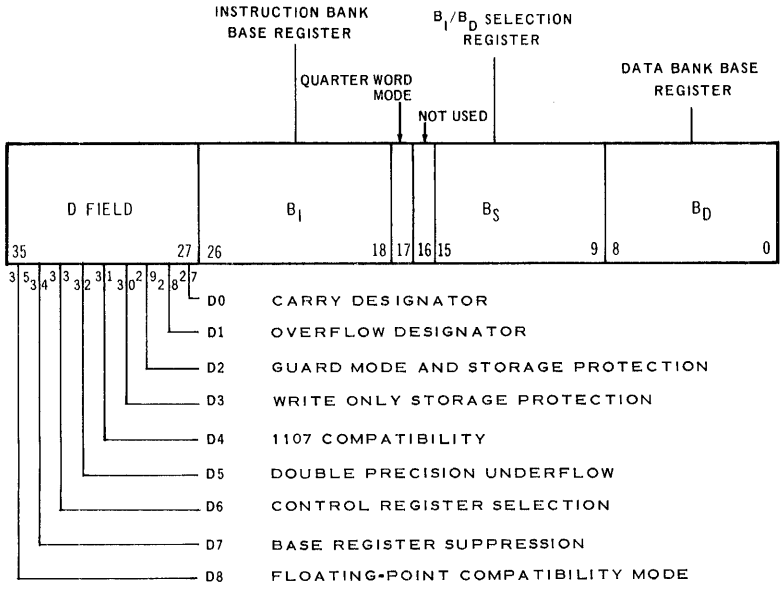


Figure 9–1. Format of the Processor State Word

9.2.1. D8 – Floating-Point Compatability Mode Designator (F–P Zero)

When D8 = 0 and the mantissa of the most significant word of a single-precision floating-point result is ± 0, the entire word is stored as all zeros.

When D8 = 1 and the mantissa of the most significant word of a single-precision floating-point result is $+0$, the most significant word is packed and stored with the appropriate characteristic (see 4.5.1).

The operations of Floating Add, Floating Add Negative, Floating Multiply, Floating Divide, and Load And Convert To Floating (which has only one result word) instructions are influenced by this designator.

9.2.2. D7 – Base Register Suppression Designator (Executive Mode)

When D7 = 0, the instruction i field specifies indirect addressing and base register indexing is used to develop main storage addresses. The absolute operand, result, or jump to address is developed in the index subsection as $(u + BI) + X_m$, or $(u + BD) + X_m$. When D7 = 0, and i = 1, the i field specifies indirect addressing, except for the cases in which f = 01 – 67_8, j = 16 or 17, and x = 0. (See 5.2.7.)

When D7 = 1, the i field is used to specify base register suppression rather than indirect addressing. When D7 = 1 and i = 1, the contents of the BI and BD fields of the PSR are ignored, and the absolute operand address, result address, or jump to address is developed in the index subsection as $(u + 0) + X_m$. The i field can still be used with the h and u fields to form an 18-bit operand for those cases in which f = 01 – 67_8, j = 16 or 17, and x = 0. When D7 = 1 and i = 0, base register indexing functions the same as for D7 = 0 (see 5.3.3.1).

The operation of the P capturing instructions, Store Location And Jump (SLJ) and Load Modifier And Jump (LMJ), when base register suppression has been specified, is discussed in 9.3.6.

9.2.3. D6 – Control Register Selection Designator (Exec ABR)

When D6 = 0, the a and x fields of an instruction word reference the addressable control registers assigned for use by the user program. These are the control registers having addresses in the ranges from 000_8 through 037_8 and 100_8 through 117_8. (See 3.3 and Table 3–6.)

When D6 = 1, the a and x fields of the instruction word reference addressable control registers assigned for use by the Executive program. These are the control registers having addresses in the range 120_8 through 177_8.

9.2.4. D5 – Double-Precision Underflow Designator (Interrupt Suppression)

When D5 = 0, a Floating-Point Characteristic Underflow Fault Interrupt occurs if characteristic underflow is detected during the execution of a double-precision floating-point instruction. The contents of the specified A registers, A and A +1, remain unchanged.

When D5 = 1, a Floating-Point Characteristic Underflow Fault Interrupt does not occur if characteristic underflow is detected during the execution of a double-precision floating-point instruction. Instead, the contents of the specified A registers, A and A + 1, are cleared to all zeros, and the normal instruction sequence is continued.

9.2.5. D4 – 1107 Compatibility Designator (1107 Mode)

When D4 = 0, U, SI, and SD are formed and used to provide a main storage absolute address range from 000000_8 to 777777_8 (262K words). See 5.3.3.1.

When D4 = 1, the absolute addresses formed are not defined unless BI = BD = BS = zero. When D4 = 1, and BI, BD, and BS are zero, the main storage absolute address range is 000000_8 to 177777_8 (65K). When the u field specifies an operand address, result address, or jump address, U, SI, and SD are, in effect, formed in the normal manner (U = SI = SD, since BI = BD = 0), and then 0 bits are forced into bits 17 and 16 of U, SI, and SD before U is tested to determine whether or not it is to be used as the address of a control register and before its comparison with BS (which is also zero). See Figure 9–2.

9.2.6. D3 – Modified Storage Protection (Write Only); D2 – Guard Mode/Storage Limits Protection

D3 and D2 are interactive. The functions of these designators are defined together in the following paragraphs.

When D3, D2 = 00_2, guard mode and storage limits protection are disabled. A Guard Mode/Storage Limits Protection Fault Interrupt will not occur for any reason.

When D3, D2 = 01_2, guard mode and storage limits protection are fully enabled. A Guard Mode/Storage Limits Protection Fault Interrupt (to MSR + 243_8) occurs when an attempt is made to execute any of the privileged instructions (listed as items 1 through 12 under 8.3.2.2), when any attempt is made to write (store) into any of the I/O Access Control Registers (040_8 through 077_8), the Real Time Clock register (100_8), or the executive registers (120_8 through 177_8), when an Execute instruction or indirect addressing loops or cascades for more than 100 microseconds, or when the main storage address developed for reading an operand, writing a result, or loading the P register for a jump operation violates the Storage Limits Register (see 9.4).

When D3, D2 = 10_2, guard mode is disabled and storage limits protection is disabled for reads and jumps. However, a Storage Limits Protection Fault Interrupt (to MSR + 243_8) occurs when the main storage address developed for writing a result violates the SLR.

When D3, D2 = 11_2, guard mode is enabled, and storage limits protection is enabled only for write operations, the guard mode protection is identical to that provided when D3, D2 = 01_2 as explained above. However, the protection provided by the SLR applies only to write operations; it does not apply to operand reads or to jump operations.

NOTE: When D2 = 1 and a Halt Jump/Halt Keys And Jump instruction is performed, the halt portion of the instruction is not performed. Instead, the instruction from the jump to location U is immediately executed.

9.2.7. D1 – Overflow Designtor

D1 is always cleared to 0 when the execution of one of the following arithmetic instructions is initiated.

Add To A

Add Negative To A

Add Magnitude To A

Add Negative Magnitude To A

Add Upper

Add Negative Upper

Add To X

Add Negative To X

Double Precision Fixed Point Add

Double Precision Fixed Point Add Negative

D1 is set to 1 if an overflow condition (see 4.3.3.1) is detected when one of the above instructions is performed.

9.2.8. D0 – Carry Designator

D0 is always cleared to 0 when the execution of one of the ten instructions listed in 9.2.7 is initiated.

D0 is set to 1 if a carry condition (see 4.3.3.2) is detected when one of the ten instructions listed in 9.2.7 is performed.

NOTE: When D1 and/or D0 is set to 1, it remains set to 1 until one of the instructions listed in the explanation of D1 is executed or until the contents of the PSR are changed as a result of some other event such as the CPU responding to an interrupt signal (see 8.1) or performing a Load Processor State instruction.

9.2.9. BI – Instruction Bank Base Register

BI is the Instruction Bank Base Register. It is used during the conversion of a relative address to an absolute main storage address. BI (or BD) is also used during the execution of an LMJ or SLJ instruction (see 9.3.6) to convert the absolute main storage address in the P register to a relative address to be stored. BI is used for this conversion when SI (or SI + 1) was used as the most recent jump to address in the program. The relative to absolute address conversion process is explained in 9.3.4.

9.2.10. QW – Quarter Word Designator

When QW = 0 and the f field of an instruction contains a value in the range 01_8 through 67_8, then i field values of 4, 5, 6, and 7 in that instruction are interpreted as follows for a main storage reference:

j = 4: Specifies half word (18-bit) transfers to or from bits 35 through 18 of the specified location.

j = 5: Specifies third word (12-bit) transfers to or from bits 11 through 0 of the specified location.

j = 6: Specifies third word (12-bit) transfers to or from bits 23 through 12 of the specified location.

j = 7: Specifies third word (12-bit) transfers to or from bits 35 through 24 of the specified location.

When QW = 1 and the f field of an instruction contains a value in the range 01_8 through 67_8, the j field values of 4, 5, 6, and 7 in that instruction are interpreted as follows for a main storage reference:

j = 4: Specifies quarter word (9-bit) transfers to or from bits 26 through 18 of the specified location.

j = 5: Specifies quarter word (9-bit) transfers to or from bits 8 through 0 of the specified location.

j = 6: Specifies quarter word (9-bit) transfers to or from bits 17 through 9 of the specified location.

j = 7: Specifies quarter word (9-bit) transfers to or from bits 35 through 27 of the specified location.

The value of the QW designator has no effect on an instruction in the following circumstances:

- when the f field of the instruction contains a value in the range 70_8 through 77_8;
- when the j field contains a value other than 4, 5, 6, or 7; or
- when U is used to address a control register. All transfers to and from control registers are always full word transfers regardless of the values of the QW designator and the j field of the instruction.

9.2.11. NU = Not Used

The value in bit 16 of the PSR is not used (that is, it has no effect on the operation of the system). However, it is recommended that bit 16 of the word sent to the PSR by a Load Processor State instruction contain a 0 bit.

9.2.12. BS = BI/BD Selection Register

BS is the BI/BD Selection Register. It is used to determine whether a relative address references an absolute main storage address in the I bank or D bank portion of main storage. The value assigned for BS should be greater than or equal to the nine most significant bits of the largest relative address in the I segment of the program and less than the nine most significant bits of any relative address in the D segment of the program. The function of BS in UNIVAC 1108 main storage addressing is explained in 9.3.4.

9.2.13. BD – Data Bank Base Register

BD is the Data Bank Base Register. It is used during the conversion of a relative address to an absolute main storage address. BD (or BI) is also used during the execution of an LMJ or SLJ instruction (see 9.3.6) to convert the absolute main storage address in the P register to a relative address to be stored. BD is used for this conversion when SD (or SD + 1) was used as the most recent jump to address in the program. The relative to absolute address conversion process is explained in 9.3.4.

9.2.14. Loading the Processor State Register

The Processor State Register can be loaded by performing a Load Processor State instruction. When this instruction is executed, the word addressed by the instruction (the Processor State Word) is transferred to the PSR. The Executive program must define a Processor State Word with the desired designators set and the proper values of BI, BD, and BS for each user program under its control. The Executive program must load the proper Processor State Word into the PSR before it transfers control to a user program. An example of the sequence of instructions which an Executive program might perform to transfer control to a user program is given in note (5) of 6.14.3.

Control is returned to the Executive program only when an interrupt occurs. The LMJ or SLJ instruction in the interrupt location transfers control to the Executive program. When an interrupt occurs, the contents of the PSR are automatically stored in the control register at address 000_8 and the PSR is then automatically forced to contain 1 bits for D7 and D6 and 0 bits for D8, D5 through D0, NU, and QW. The values for BI, BD, and BS in the PSR are not automatically changed when an interrupt occurs. If these values need to be changed, the Executive routine handling the interrupt must set them to the desired values by performing a Load Processor State instruction.

9.3. INTRODUCTION TO ADDRESSING

The CPU's addressing hardware provides for relocating the instructions and/or the data for any program in main storage. The hardware design provides the ability to specify two areas of memory for use by a running program. The two areas are referred to as segments. The design also provides the ability to specify that all areas of main storage not assigned to a program are locked out to that program for both read and write (or write only) references. The areas of main storage which may be assigned to a program can be specified in gradations of 512 words – thus, an area may be opened up for use, that is, 512 words, or 1024 words, or 1536 words. An area of 512 words is referred to as a block of main storage.

9.3.1. Main Storage Organization

The UNIVAC 1108 is designed as a modular system, permitting a variety of main storage configurations. For noninterleaved main storage, the minimum configuration consists of two 32K modules (65K). This system can be expanded to a maximum of 131K words. For an interleaved main storage, sequentially addressed main storage words within a 65K module pair are alternately located in the two modules. Two module pairs (131K words) are the minimum complement of interleaved main storage available, and this configuration may be expanded in increments of 65K words to a maximum complement of 262K words.

The base registers, BI and BD, which provide the CPU with the flexibility for relocating programs are 9-bit registers. The Storage Limits Register contains four 9-bit fields which are used to specify the areas of main storage assigned to a specific program. The use of 9-bit registers allows a maximum main storage configuration of 262K (2^{18}) words to be divided into 512 (2^9) blocks of 512 words each, or 1000_8 words. Any location within a block of main storage can be addressed by using values of 000_8 through 777_8. The block address and the address of the location within a block can be combined to address any location from 000000_8 through 777777_8. Table 9–1 shows the ranges of block numbers for noninterleaved modules and interleaved module pairs of main storage.

NONINTERLEAVED MAIN STORAGE		INTERLEAVED MAIN STORAGE	
MAIN STORAGE MODULE NUMBER	BLOCK NUMBERS	MAIN STORAGE MODULE PAIR NUMBER	BLOCK NUMBERS
Module #0 (Mem 1)	$000\text{-}077_8$	Module Pair #0 (Mem 1)	$000\text{-}177_8$
Module #1 (Mem 2)	100-177	Module Pair #1 (Mem 3)	200-377
Module #2 (Mem 3)	200-277	Module Pair #2 (Mem 2)	400-577
Module #3 (Mem 4)	300-377	Module Pair #3 (Mem 4)	600-777

Table 9–1. The Range of Block Numbers for Noninterleaved and Interleaved Main Storage

9.3.2. Program Segmentation

A program can be written in two segments or portions, each of which may be relocated in main storage. The segment having the lower relative address is the I segment, and the segment having the higher relative address is the D segment. If the instructions for the program are in one segment and the data is in the other segment, the Executive program can attempt to assign the segments to different banks of main storage so that alternate bank timing will apply to the instruction execution.

When a program is loaded into main storage, the Executive must determine how many blocks are required for each segment based on the segment size. Then it can assign each segment of the program to main storage blocks not assigned previously to other programs. An individual segment is loaded into contiguous blocks starting with the first location within the first block assigned to that segment. The segment may not entirely fill all the storage locations in the last block assigned to it. The unfilled portion of that block is ordinarily not used if the program is to be run with guard mode/storage limits protection because the granularity for memory protection is 512 words.

9.3.3. General Theory of 1108 Addressing

Normal 1108 programs are constructed without consideration for the physical area of main storage they will occupy during execution. As the program is constructed, each word is mapped into a set of addresses called relative addresses. A relative address is actually used in an instruction within the program which references other locations or words in the program. Proper conversion from these relative addresses to the physical locations of the program will occur during execution using the base register mechanism of the 1108.

The range of relative addresses is from 0 to 262,143. Three constraints limit the portions of the relative address range into which programs are mapped.

- Addresses 0 through 127 always reference control registers and are unavailable for main storage access.
- The additional difficulty in forming addresses greater than 65,535 makes it desirable when possible to map the program within the first 65,536 addresses.
- Execution of instructions located at addresses greater than 65,535 require special addressing considerations. (See 9.3.5 and 9.3.6.)

If a program was mapped beginning at relative address 0, the first 128 words of the first block assigned in memory could not be referenced. A programming convention which maps addresses beginning at 200_8 or greater avoids this waste. For convenience, beginning addresses at multiples of 1000_8 is desirable as they correspond with the block assignments for the program. Physical space need not be assigned corresponding to the relative addresses below the minimum address. The Storage Limits Register can be adjusted to prevent reference in this range.

A program consists of one or two segments. Normally two are prepared to take advantage of the overlapped main storage references possible in the 1108 which provide faster execution. Two base registers are available for independent relocation of each segment. A part of the relative address range is assigned to reference each of the segments. The I segment is always referenced with the smaller relative address values; the D segment with the larger relative addresses. The constraints given above in selecting the relative address map apply to either case. However, the relative address range used for reference to each of the two segments need not be contiguous.

The Base Selection register (BS) defines for the addressing algorithm the split in addresses between the two segments. BS remains fixed for each constructed program, and is unaffected by the actual placement in main storage of the program. BS is a 7-bit field of the Processor State Register which points to the highest block of the I segment. The largest address BS can represent is 0177777_8 or 65,535. Any address reference greater than this value will always be a D segment reference. The I segment is limited to this length. If one imagines the BS register extended on the left by two 0 bits and on the right by nine 1 bits, then:

- A relative address is an I segment reference if less than or equal to the extended BS.
- Any relative address greater than the extended BS is a D segment reference.

The I segment and D segment will typically be placed in noncontiguous main storage areas. If relative addresses are mapped continuously at the break, references during execution will appear continuous across the physical break with one important exception. Each jump instruction loads into the P register an absolute address constructed from the relative address supplied by the program. Sequential instruction execution proceeds by incrementing the absolute P register value by 1. If the two segments are not physically contiguous, and sequential instruction execution proceeds through the last word of the I segment, the next instruction will not be secured from the D segment but rather from the first absolute location following the physical I segment. Normal storage protection applied to user programs would cause a guard mode fault.

The base registers BI and BD are the relocation factors applied to each program address reference to adjust for the physical position of the program in main storage. BS is compared with each relative address to determine which of the two base registers is added in forming the absolute address. BI and BD are 9-bit fields in PSR. Each can be thought of as an address by extending it on the right with nine 0 bits.

A free area of sufficient size will be selected by the Executive and the I segment loaded into it. BI is calculated as the difference between the absolute address of any word in the I segment and the relative address assigned by the program to this location. For example, by programming convention the first word of the I segment is assigned relative address 1000_8. If absolute address $70,000_8$ is selected as the first physical location of the I segment, the BI address value is easily calculated as ($70,000_8 - 1000_8$) or $67,000_8$. The Executive truncates and assigns this BI value to the Processor State Word for the program.

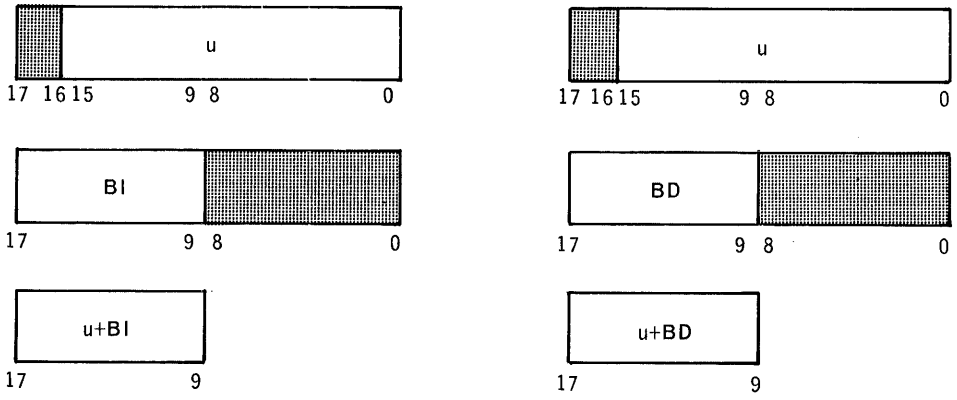
A similar selection of physical space to hold the D segment and calculation of the value for BD must be carried out by the Executive. One additional constraint exists in the selection of physical space for the D segment. The addressing algorithm operates properly only on positive values. BD must be equal to or greater than 0. This requires that the absolute address of any D segment word be equal to or greater than the relative address by which this word is known. This same constraint also applies to the I segment, but is usually of no concern as the smallest convenient relative addresses are typically used for the I segment.

The T0 timing chain is also used to secure the operands for repeated instructions such as the Block Transfer. I/O service requests are satisfied during the long duration instructions by breaking into the sequence and inserting a T8 timing chain cycle at the point in time at which the next T0 cycle would have been initiated.

9.3.4. Description of the Base Register Addressing Process

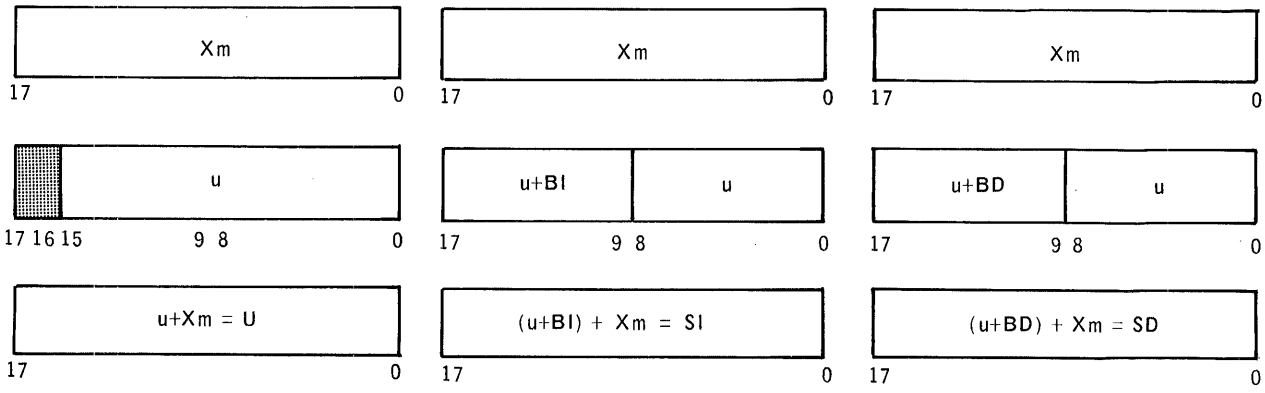
The flowchart in Figure 9–2 illustrates the effective sequence of operations during the formation of an address. The equivalent of five additions is performed.

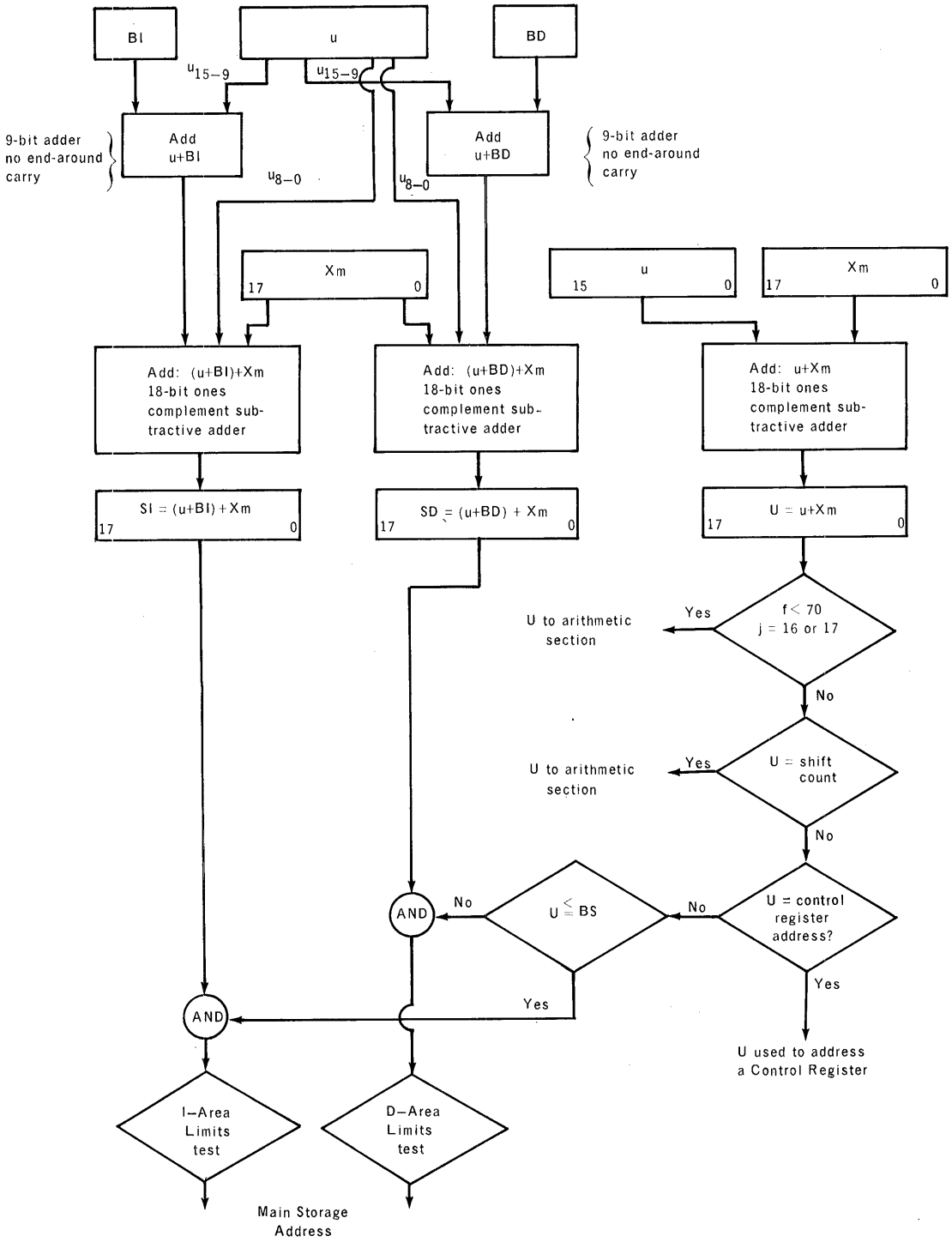
- At the start of the process, the sums u+BI and u+BD are formed in parallel. The alignment for these sums is as follows:



These are 9-bit additions with no end-around carry. If there is overflow on the addition, the 10th bit is discarded. (See 9.3.5.)

- After the contents of the X register have been read out, the following three sums are formed in 18-bit registers.





NOTES: (1) The next consecutive main storage address must also be developed for double-precision instructions and for the SLJ instruction. In these cases, the hardware forms the second address by clearing the register holding u and sending U+1 to the register holding Xm. This means that SI+1 and SD+1 are formed as follows:

$$SI+1 = (BI+0) + (U+1)$$

$$SD+1 = (BD+0) + (U+1)$$

(2) When D7 of the Processor State Word is 1, the i field can be used to specify base register suppression for instructions which require developing a main storage address. When D7 = 1 and i = 1, the absolute main storage address developed in the index subsection is (0+u) + Xm. (See 9.3.6.)

Figure 9–2. UNIVAC 1108 Addressing Operation Flowchart

These sums are formed in ones complement 18-bit subtractive adders which function as described in 4.2.

- If the f, j, and x fields of the instruction call for U to be used as an 18-bit operand, then U is sent to the arithmetic section.
- If the instruction is a shift instruction, then U is sent to the arithmetic section to be used as a shift count.

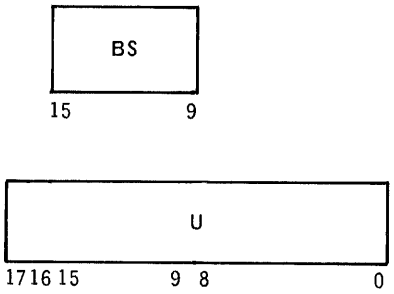
Otherwise, U is examined to determine whether or not it is a control register address or a main storage address.

- If U is a control register address, it is used to reference a control register.

If it is not, then U is compared with BS to determine whether SI or SD should be referenced.

- If U $\leq$ BS, the main storage address referenced is SI.
- If U $>$ BS, the main storage address referenced is SD.

The comparison of BS with U can be illustrated more clearly by first designating the seven bits of BS as 15, . . . , 9. The bits of U that are tested with BS are aligned as follows:



During the comparison, U_{17} and U_{16} are sampled. If either or both is a 1, SD is automatically taken as the main storage address referenced. Otherwise, BS is compared with U_{15-9} to determine which address is taken.

9.3.5. Programming Considerations Related to Addressing

The hardware which performs the relative to absolute address conversion was designed to function properly provided that the relative address is always less than or equal to the absolute address. This requirement can be stated briefly as:

- If SI is the main storage address to be referenced ($U_{17-9} \leq BS$), then the hardware requires that $U \leq SI$.
- If SD is the main storage address to be referenced ($U_{17-9} > BS$), then the hardware requires that $U \leq SD$.

An Executive program can ensure that these requirements are met by selecting the values for BI and BD such that

$$U_{17-9} + BI \leq 777_8,$$

and

$$U_{17-9} + BD \leq 777_8,$$

where U in the first case is the largest relative address which can occur in the I segment of the program, and U in the second case is the largest relative address which can occur in the D segment of the program.

These requirements stem from the fact that the sums

$$u_{15-9} + BI \text{ and } u_{15-9} + BD$$

are formed in 9-bit adders. If $u_{15-9} + BI > 777_8$ or $u_{15-9} + BD > 777_8$, overflow has occurred and the leading bit of the 10-bit sum cannot be contained in the 9-bit register which holds the sum.

The absolute address generated in these cases appears to be smaller than the relative address. For example, if

$$u = 020000_8,$$

$$BD = 770_8, \text{ and } Xm = 0,$$

then

$$u_{15-9} + BD = 020 + 770 = 1010$$

However, only nine bits of this sum are retained so that value appears as

$$u_{15-9} + BD = 010$$

Then the relative address $U = u + Xm = 020000_8$ and the absolute address $SD = (u + BD) + Xm = 010000_8$.

For double-precision instructions and for the SLJ instruction, the next consecutive main storage address must be referenced. In these cases, the next consecutive address SI + 1 or SD + 1 is formed as (BI + 0) + (U + 1) or (BD + 0) + (U + 1) where U = u + Xm. The main storage address developed for either SI + 1 or SD + 1 must be less than or equal to 777777_8. When this restriction is violated, the absolute main address developed in the index subsection is two greater than the corresponding SI or SD.

In the previous example

$$BD + 0 = 770000_8$$

$$U + 1 = 020001_8$$

$$SD + 1 = (BD + 0) + (U + 1) = 010002_8$$

Thus we have $SD = 010000_8$

and $SD + 1 = 010002_8$

The two addresses developed are not consecutive because an overflow bit was produced (and ignored) when the 9-bit sum u_{15-9} + **BD** was formed; but when the 18-bit sum (BD + 0) + (U + 1) was formed, the overflow bit generated an end-around carry which was added into the low order bit position.

This problem will not arise in normal circumstances where no overflow occurs in either u_{15-9} + BI (if SI is selected as the absolute main storage reference) or u_{15-9} + BD (if SD is selected as the main storage reference), and either SI or SD is less than 777776_8 for those operations which require the next consecutive address.

A similar problem can arise if the value for Xm is negative (i.e., if the value for Xm has leading 1 bits). In this case the value assigned to BI and BD (whichever is associated with the relative address U) must not be so great that BI + u_{15-9}>777 (or BD + u_{15-9}>777).

This problem is avoided when addressing locations within a segment by not assigning values to u which exceed the largest relative address for that segment.

9.3.6. P-Capturing Instructions

The Program Address Register (P register) contains the absolute main storage address of the next instruction to be read from main storage and loaded into the CPU's control section for the normal sequential execution of a list of instructions. When one of the two P-capturing instructions, Store Location And Jump (SLJ) or Load Modifier And Jump (LMJ) is executed, BI or BD is subtracted from the contents of the nine leftmost bit positions of the P register to obtain the relative address which corresponds to the absolute main storage address in the P register. This relative address is stored by the SLJ or LMJ instruction.

Whenever any jump instruction is performed, the CPU records which of the two values, SI or SD (SI + 1 or SD + 1 in the case of an SLJ instruction) was used as the absolute jump to address. By updating this record each time a jump occurs, the CPU maintains a record which indicates whether the absolute main storage address in the P register is associated with BI or BD. When an SLJ or LMJ instruction is performed, the CPU determines the relative address to be stored by subtracting either BI or BD from the contents of bits 17 through 9 of the P register. If SI or SI + 1 was last used as the absolute jump to address, BI is subtracted; if SD or SD + 1 was last used, BD is subtracted. When BI or BD is subtracted from the absolute main storage address in the P register to form the relative address to be captured by the SLJ or LMJ instruction, any end-around borrow generated during the subtraction is suppressed.

When a program is operating with base register suppression (D7 = 1 and i = 1), the base register suppression applies to each absolute main storage address developed using the value in the u field, but not to the captured relative address derived from the contents of the P register. Base register suppression applies to the calculation of the absolute main storage address at which the captured relative address is stored for the SLJ instruction. The jump to addresses for the LMJ and SLJ instructions are also calculated with base register suppression. The jump to address for the LMJ instruction is developed according to the procedure in Figure 9–2 except that BI and BD are effectively zero so that $(u + BI) + X_m$ and $(u + BD) + X_m$ reduce to $u + X_m$. The jump to address for the SLJ instruction is developed according to the procedure described in Note (1) for Figure 9–2 except that BI and BD are effectively zero so that (BI + 0) + U + 1 and (BD + 0) + U + 1 reduce to U + 1.

The LMJ and SLJ instructions are frequently used in the interrupt locations to transfer control from a user program to an interrupt handling routine. D7 is automatically set to 1 when the interrupt occurs so that if the LMJ or SLJ specifies i = 1, base register suppression will apply to the jump to address (and to the address specified by the SLJ instruction for storing the user jump from address). The user jump from address is a relative address formed by subtracting BI or BD from the absolute main storage address in the P register. If either an LMJ or SLJ instruction is performed at some point within the interrupt handling routine, the captured address will be a relative address. Thus the interrupt handling routine must clear BI and BD if these instructions are to be used to capture the proper address within a routine operating with base register suppression. The PSR must be restored to the appropriate values for the user before control is returned to that user. (See 9.2.14.)

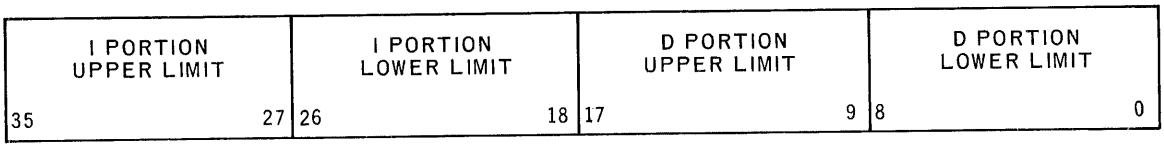
If the SLJ instruction is used to capture the relative jump from address and transfer control to another sequence of instructions, the procedure for returning to the first sequence of instructions is simplified if the relative value captured is less than 200000_8. If the relative value captured is 200000_8 or greater, it contains a 1 bit in bit 17 or 18 (or both). If this relative address is used as the right half of an instruction, any 1 bits in these positions will be interpreted for index register incrementation and indirect addressing (or base register suppression) rather than as bits used in developing an absolute address. However, if all the instructions for a program have relative addresses of 177777_8 or less (the largest possible relative address in the I bank 177777_8), this situation will not arise (see 6.9.1, Note 7).

9.4. MAIN STORAGE PROTECTION

The CPU has the capability of specifying two areas of main storage for use by a running program. All locations in main storage not assigned to a program can be locked out to that program either for both read and write references or only for write references. The portions of main storage assigned to a program are specifiable in blocks of 1000_8 words (see 9.3.1). There are two sets of storage limits, one for each of the two areas of main storage which may be assigned to a program. These storage limits are contained in the Storage Limits Register (SLR). The upper and lower absolute address limits for the I segment and the D segment of a program are defined in the SLR. These limits are expressed as block numbers. The two segments of main storage defined by the SLR may be completely separate segments or partially overlapping segments, or one segment may be a subset of the other.

9.4.1. Format for the Storage Limits Word

The Storage Limits Word contained in the SLR has the format shown below. The upper half of the SLR is used to define the upper and lower limits of the I segment of a program. The lower half of the SLR is used to define the upper and lower limits of the D segment of the program.



9.4.2. Loading the Storage Limits Register

The SLR is loaded by the Load Storage Limits instruction (LSL). This is the only way the SLR can be loaded. The execution of this instruction transfers the contents of location U to the SLR, but it does not enable main storage protection.

9.4.3. Activating and Deactivating Main Storage Protection

Main storage protection is activated when a Load Processor State instruction (LPS) is executed which sets D3D2 (bits 30 and 29) of the Processor State Word to one of the three types of guard mode/storage limits protection (see 9.2.6). Main storage protection is deactivated when D3D2 of the Processor State Word are cleared to 00_2. These two bits are automatically cleared to zero whenever an interrupt occurs. (This includes the interrupts which occur when the Executive Return instruction and the Test And Set instruction are performed). D3D2 of the PSR can also be cleared to zero by performing an LPS instruction when D3D2 = 10_2. The contents of the SLR are not disturbed when main storage protection is deactivated.

Conversion of the relative addresses (U and U + 1) to absolute main storage addresses (SI and SI + 1, SD and SD + 1) is performed in the index subsection. When either SI or SI + 1 is chosen as the absolute main storage address to be referenced, it is checked against the set of limits for the I portion of the program. When either SD or SD + 1 is chosen, it is checked against the set of limits for the D segment of the program (see Figure 9–2). In either case, the reference is permitted if it is not outside the limits specified in the corresponding part of the SLR, that is, if

$$\text{I Portion Lower Limit} \leq \text{SI} \leq \text{I Portion Upper Limit}$$

or

$$\text{D Portion Lower Limit} \leq \text{SD} \leq \text{D Portion Upper Limit}$$

When the absolute main storage address is outside the limits specified by the SLR and main storage protection has been specified for the type of operation involved, then the main storage reference is not made. Instead a Guard Mode/Storage Limits Protection Fault Interrupt occurs and the CPU executes the instruction at MSR + 243_8 as its next instruction.

The two classes of protection, read-write-jump and write only, are discussed in 9.2.6. It should be noted that the SLR in conjunction with the contents of the PSR does not provide main storage protection against the reading of sequential instructions whose addresses are formed by the normal process of incrementing the P register. Thus it is possible for a program to move along sequentially from an assigned main storage area into an area which violates the SLR. However, if guard mode/storage limits protection is fully enabled (D3D2 = 01_2), then main storage protection is provided for any jump to address which violates the SLR.

APPENDIX A. SYMBOLS AND ABBREVIATIONS

a	The a field (bits 25-22) of an instruction.
A	Control register specified by the a field of an instruction.
A register	A control register (addresses 14_8 – 33_8 and 154_8 – 173_8). Registers at addresses 34_8, 35_8, 174_8, and 175_8 can be used either as general purpose registers or as extensions of an A register.
Aa	The A register specified explicitly by the a field of an instruction.
Aa+1	An A register whose address is one greater than the control register address specified by the a field of an instruction.
Aa+2	An A register whose address is two greater than the control register address specified by the a field of an instruction.
ACU	Availability Control Unit
ACW	Access Control Word
AND	Logical product
A+1	Same as Aa+1
A+2	Same as Aa+2
BD	The BD field (bits 8 – 0) of the Processor State Register. BD times 2^9 yields the address of the first word in the D-bank.
BI	The BI field (bits 26 – 18) of the Processor State Register. BI times 2^9 yields the address of the first word in the I-bank.
BS	The BS field (bits 15 – 9) of the Processor State Register. BS times 2^9 plus 777_8 yields the address of the last word in the I-bank.
C	The C field (bits 31 – 30) of an Access Control Word for a Quarter Word ESI channel.

Characteristic	Biased exponent portion of a floating-point number.
CLT	Communication Line Terminal
CPU	Processor Unit (the computer itself as opposed to the I/O Controller which, in the UNIVAC 1108 Multi-Processor System, is also considered to be a processor)
CSR	Channel Select Register
CTM	Communication Terminal Module
CTMC	Communication Terminal Module Controller or Communication Terminal Module Control Subsystem, dependent upon context.
D-bank or D-portion	One of the two areas in main storage assigned to the program. The area which corresponds to the higher relative addresses. See also I-bank.
D0	The carry indicator (bit 27) in the Processor State Register.
D1	The overflow indicator (bit 28) in the Processor State Register.
D2	The guard mode/storage limits protection indicator (bit 29) in the Processor State Register. Used to specify guard mode and whether a Guard Mode Fault Interrupt should occur if an attempt is made: (1) to perform any restricted instruction; (2) to write in a main storage location outside the area specified by the Storage Limits Register; or (3) to write in any restricted control register. It is also used to prevent a program halt from occurring for a halt-type instruction, and to determine whether the contents of the Storage Limits Register are to be used in restricting reads and jumps when D3 = 0.
D3	The modified storage limits protection (write only) indicator (bit 30) of the Processor State Register. Used to specify whether the Storage Limits Register only writes (D3 = 1) or is to be used in conjunction with D2 = 1 to restrict reads, writes, and jumps (D3 = 0).
D4	The 1107 compatibility indicator (bit 31) of the Processor State Register. Used to specify 1107 address compatibility, that is, whether the leftmost two bits of the 18-bit address produced and selected in the index subsection are to be left unchanged or cleared to 0 bits just prior to sending the address to the main storage address circuits.
D5	The double-precision underflow indicator (bit 32) of the Processor State Register. Used to specify whether the result of characteristic underflow should be the initiation of a Characteristic Underflow Fault Interrupt or the storing of the result as all 0 bits.

D6 — The control register selection indicator (bit 33) of the Processor State Register. Used to specify which of X, A, or R registers is to be used.

D7 — The base register suppression indicator (bit 34) of the Processor State Register. Used to specify the interpretation of the i field in an instruction. D7 = 1 selects absolute addressing; D7 = 0 enables either indirect addressing or u field extension, as appropriate.

D8 — The floating-point compatibility mode indicator (bit 35) of the Processor State Register. Used to specify form of special treatment to be given to word stored in Aa for a single-precision floating-point operation when the sign/mantissa arithmetic produces a result of all 0 bits or all 1 bits.

EF — External Function. Control signal sent by CPU to a subsystem which identifies the word on the output data lines as a function rather than a data word.

EI — External Interrupt. Control signal sent by a subsystem to the CPU which causes an interrupt and identifies the word on the output data lines as a status rather than a data word.

ESI — Externally Specified Index. Applied to a subsystem and the I/O channel connecting that subsystem to the CPU. An ESI subsystem specifies a relative address when it requests an input or output data transfer. This relative address is converted by the CPU to an absolute address. The absolute address is used as an address of an Input or Output Access Control Register rather than a control register.

f — The f field (bits 35 – 30) of an instruction; the function code.

f,j — The f and j fields (bits 35 – 26) of an instruction. When the function code is greater than 70_8, the f,j combination defines the basic operation to be performed.

F0 — A 36-bit register in the program control subsection which receives each instruction read out of main storage and the low order 22 bits of the word read out of main storage during an indirect addressing sequence.

F1 — A 14-bit register in the program control subsection used to store the f and j fields of an instruction and an OR CSR (logical sum of a field and Channel Select Register).

F3 — A 7-bit register used to store the address of a control register.

F4 — A 7-bit register used to store the address of an A register when an instruction references two A registers.

G — The contents of bit positions 35 and 34 (G field) of an Input or Output Access Control Word. For each I/O transfer, G specifies address incrementation (g = 00), decrementation (g = 10), or no change (g = 01).

h — The h field (bit 17) of an instruction. Normally, this field specifies whether or not the index register is to be incremented. In some instances, it is used to extend the u field.

H	A one- or two-bit field (bit 33 or bits 33 and 32) of an Input or Output Access Control Word. It is used to specify which portion (half or quarter) of a word is to be transferred.
i	The i field (bit 16) of an instruction. It normally specifies indirect addressing; however, in some instances, it is: (1) used to extend the u field; or (2) used in conjunction with D7 to specify base register suppression.
I-bank or I-portion	One of two areas in main storage assigned to a program. The area which corresponds to the lower relative addresses.
IA	Input Acknowledge. Control signal sent by the CPU to the subsystem in response to an Input Data Request or External Interrupt signal and it indicates that the CPU has accepted the input status or data word.
IACR	Input Access Control Register. For an Internally Specified Index I/O channel, the IACR is one of the control registers at addresses $40_8 - 57_8$. For an Externally Specified Index I/O channel, the IACR is a main storage location associated with an input device.
IACW	Input Access Control Word
IDR	Input Data Request. Control signal sent from a subsystem to the CPU when the subsystem has data for input to the CPU.
IFR	Internal Function Register. Another name for the Processor State Register.
Increment	The leftmost 18 bits (Xi field) of an index register.
Incrementation	The addition of the 18 leftmost bits (Xi field) of an index register to its rightmost 18 bits (Xm field). When incremented in this manner, the resultant value in the Xm field may be greater than, less than, or equal to, its original value.
I/O	Input and Output
IOC	I/O Controller
ISI	Internally Specified Index. Applied to a subsystem and the I/O channel connecting that subsystem to the CPU. When transferring data, the CPU's initial response is to read out the Input or Output Access Control Register so as to determine the main storage location of the next word to be transferred.
j	The j field (bits 29 – 26) of an instruction. This field is used as either an operand qualifier, a partial control register address, or a minor function code.
ja	Bit positions 29 – 22 of the Jump Greater And Decrement instruction which are used to specify a control register address.

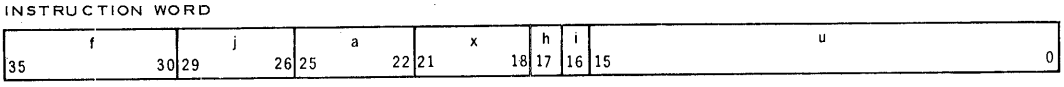
K	The initial count in the Repeat Count register for Block Transfer, search, or masked search instruction.
Ka	The console key specified by the a field of certain instructions.
LAR	Last Address Register
Mantissa	The fractional part of a floating-point number
MC	Master Clear. See Subsystem Clear.
MEM	Main Storage module
MMA	Multi-Module Access Unit
Modifier	The rightmost 18 bits (Xm field) of an index register. It is added to the 16-bit address in the u field of an instruction to produce a relative address.
MPA	Multiple Processor Adapter. Identical to Shared Peripheral Interface; see SPI.
MSR	Memory Select Register
NI	Next Instruction
Normalize	To normalize a floating-point number, the mantissa is shifted left or right until the leftmost bit of the mantissa is not identical to the sign bit.
NU	Not used. Bit position 16 of the Processor State Register.
OA	Output Acknowledge. Control signal sent by the CPU in response to an Output Request signal from a peripheral subsystem.
OACR	Output Access Control Register
ODR	Output Data Request. Control signal sent by peripheral control unit to the CPU to indicate that the control unit can accept a function or output data word.
OR	Logical inclusive OR
P	Program Address Counter (P register)
Pack	Process of combining the sign and mantissa of a floating-point number from one register with the characteristic of the number from another register into a single register.
Pass	When each of the five cycles initiated for the main timing chain (0.625 microseconds minimum) have been completed, a *pass* through the chain has been completed.
P register	Program Address Counter
PSR	Processor State Register

QW	The quarter word indicator (bit 17) in the Processor State Register. Specifies alternate interpretation of partial word designators in order to provide quarter word operations.
R or R register	Control register specified explicitly or implicitly by an instruction. Control register addresses 100_8 – 117_8 and 120_8 – 137_8.
Ra	The R register specified by the a field of an instruction.
Residue	The least significant result word produced by a single-precision Floating Add or Floating Add Negative instruction.
RTC	Real Time Clock
R0	Real Time Clock register at control register address 100_8. It is also a control register at address 120_8.
R1	Repeat Count registers at address 101_8 and 121_8. They are used during Block Transfer, search, and masked search instructions.
R2	Mask registers at addresses 102_8 and 122_8. They are used during masked search instructions and the Masked Load Upper instruction.
S	Sign bit or bit position
SCCS	Storage Class Control Subsection
SD	The absolute address (D-bank) developed through addition: (BD + u) + (Xm or 0).
SI	The absolute address (I-bank) developed through addition: (BI + u) + (Xm or 0).
SLR	Storage Limits Register
SPI	Shared Peripheral Interface (formerly called the Multiple Processor Adapter).
Subsystem Clear	Control signal sent by the CPU to each peripheral subsystem which clears the peripheral control unit and conditions it to receive the next function. From the standpoint of the peripheral control unit, it is a Master Clear signal.
T0	Main timing chain
T1	A timing chain that operates in parallel with the main timing chain which may store up to three words (from arithmetic section) in a control registers.
u	The u field (bits 15 – 0) of an instruction.
U	The 18-bit value produced in the index subsection by adding the rightmost 18 bits (Xm field) of the index register specified by the x field of the instruction (or by adding 0 if x = 0) to the 16-bit value in the u field of the instruction (u field is extended to 18 bits). This value represents an address in main storage.

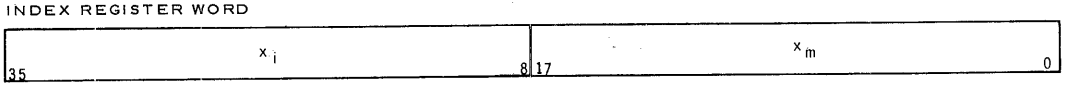
U+1	The 18-bit value produced in the index subsection by adding 1 to U. This value represents an address in main storage.
Unpack	The process of separating a floating-point number. The sign and mantissa are placed in one register and the characteristic in a different register.
V	The absolute address contained in bits 17 – 0 of an ISI or ESI Access Control Word.
W	The count field of an Access Control Word. For ISI operations, the W field is bits 33 – 18. For half word ESI operations, the W field is bits 32 – 18. For quarter word ESI operations, the W field is bits 29 – 18.
x	The x field (bits 21 – 18) of an instruction.
X or X register	Control register specified by an instruction at addresses 1_8 – 17_8 and 141_8 – 157_8. The registers at 0_8 and 140_8 are special registers.
Xa	The X register specified by the x field of an instruction.
Xi	Bits 35 – 18 of an index register. Used to increment or decrement the contents of bit positions 17 – 0 (Xm field) when specified by an instruction.
Xm	Bits 17 – 0 of an index register. Used in the index adder for the process u+Xm = U.
Xx	The X register specified by the x field of an instruction.
XOR	Logical exclusive OR
+0	Two words, one word, or a field consisting of all 0 bits.
–0	Two words, one word, or a field consisting of all 1 bits.
( )	The contents of the register or location identified by the symbol within the parentheses.
( )'	The ones complement of the register or location identified by the symbol within the parentheses.
$( )_5$	The contents of bit position 5 of the register or location identified by the symbol within the parentheses.
$( )_{17-0}$	The contents of bit positions 17 through 0 of the register or location identified by the symbol within the parentheses.

APPENDIX B. UNIVAC 1108 WORD FORMATS

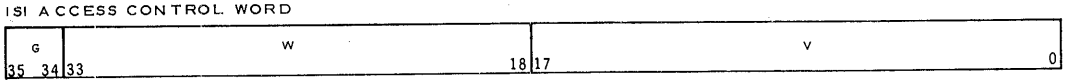
INSTRUCTION WORD



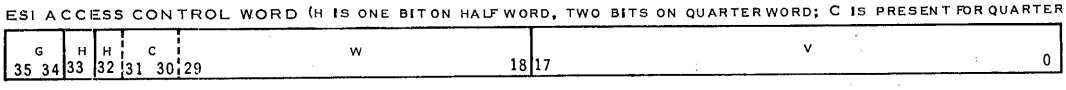
INDEX REGISTER WORD



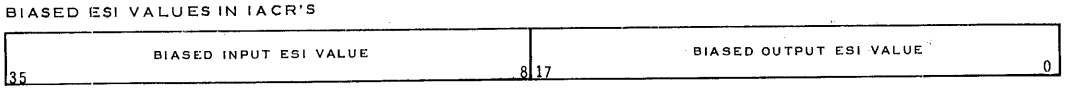
ISI ACCESS CONTROL WORD



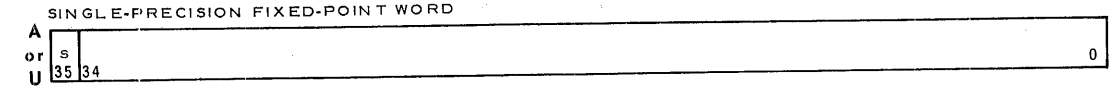
ESI ACCESS CONTROL WORD (H IS ONE BIT ON HALF WORD, TWO BITS ON QUARTER WORD; C IS PRESENT FOR QUARTER WORD)



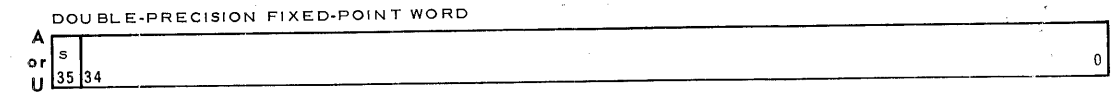
BIASED ESI VALUES IN IACR'S

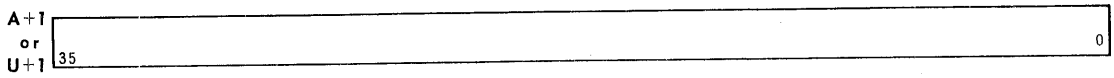


SINGLE-PRECISION FIXED-POINT WORD

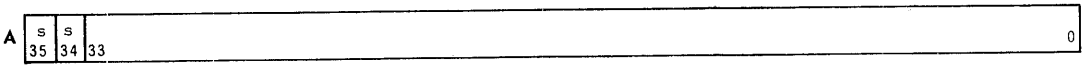


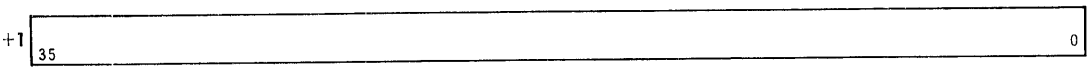
DOUBLE-PRECISION FIXED-POINT WORD



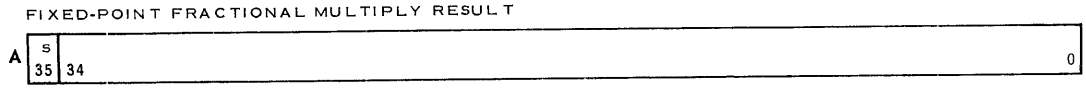


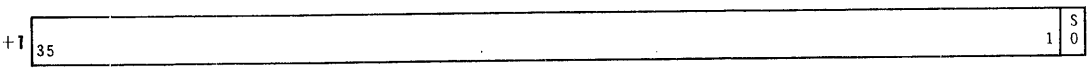
FIXED-POINT INTEGER MULTIPLY RESULT



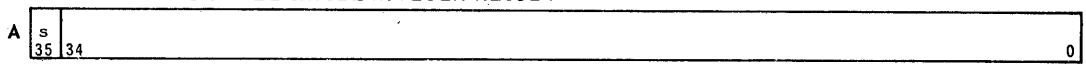


FIXED-POINT FRACTIONAL MULTIPLY RESULT

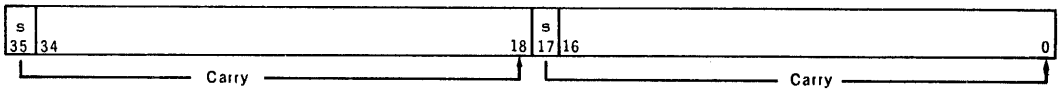




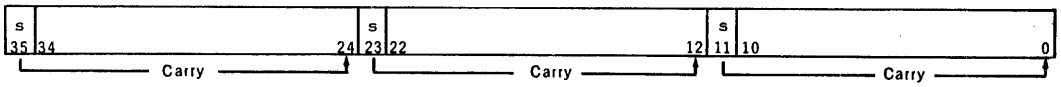
FIXED-POINT MULTIPLE SINGLE INTEGER RESULT



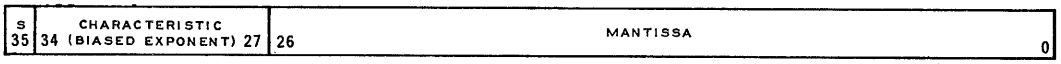
ADD HALVES WORD FORMAT



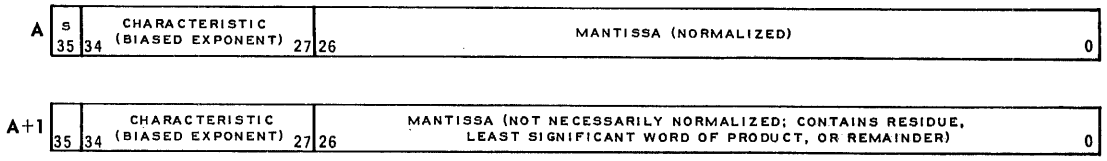
ADD THIRDS WORD FORMAT



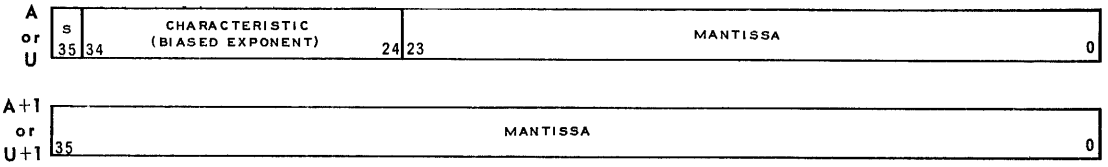
SINGLE-PRECISION FLOATING-POINT OPERAND



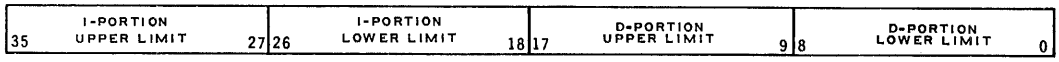
SINGLE-PRECISION FLOATING-POINT RESULT



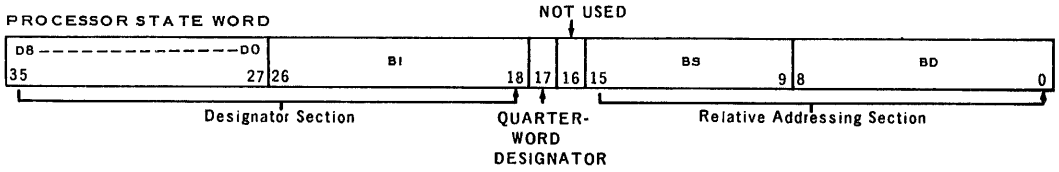
DOUBLE-PRECISION FLOATING-POINT OPERAND OR RESULT



STORAGE LIMITS WORD



PROCESSOR STATE WORD



APPENDIX C. CHARACTER CODES

CPU CODE (OCTAL)	80-COLUMN CARD CODE	HIGH SPEED PRINTER SYMBOL	DISPLAY CONSOLE TYPE 4009			OPERATOR'S CONTROL CONSOLE TYPE 4004	
			KEYBOARD SYMBOL	CRT SYMBOL	PAGEWRITER SYMBOL	KEYBOARD SYMBOL	PRINTER SYMBOL
00	7-8	@	∬	Note 1	Note 1	K	∩
01	12-5-8	[	[	[	[	UC	↑
02	11-5-8	]	]	]	]	LC	▨
03	12-7-8	#	NL (new line)	Note 2	Note 3	LF	LF
04	11-7-8	Δ	(no key)	Note 1	Note 1	RETURN	CR
05	(blank)	(space)	(space bar)	(blank)	(blank)	(space bar)	(space)
06	12-1	A	←	←	←	→	A
07	12-2	B	←	←	←	→	B
10	12-3	C	←	←	←	→	C
11	12-4	D	←	←	←	→	D
12	12-5	E	←	←	←	→	E
13	12-6	F	←	←	←	→	F
14	12-7	G	←	←	←	→	G
15	12-8	H	←	←	←	→	H
16	12-9	I	←	←	←	→	I
17	11-1	J	←	←	←	→	J
20	11-2	K	←	←	←	→	K
21	11-3	L	←	←	←	→	L
22	11-4	M	←	←	←	→	M
23	11-5	N	←	←	←	→	N
24	11-6	O	←	←	←	→	O
25	11-7	P	←	←	←	→	P
26	11-8	Q	←	←	←	→	Q
27	11-9	R	←	←	←	→	R
30	0-2	S	←	←	←	→	S
31	0-3	T	←	←	←	→	T
32	0-4	U	←	←	←	→	U
33	0-5	V	←	←	←	→	V
34	0-6	W	←	←	←	→	W
35	0-7	X	←	←	←	→	X
36	0-8	Y	←	←	←	→	Y
37	0-9	Z	←	←	←	→	Z

CPU CODE (OCTAL)	80-COLUMN CARD CODE	HIGH SPEED PRINTER SYMBOL	DISPLAY CONSOLE TYPE 4009			OPERATOR'S CONTROL CONSOLE TYPE 4004	
			KEYBOARD SYMBOL	CRT SYMBOL	PAGEWRITER SYMBOL	KEYBOARD SYMBOL	PRINTER SYMBOL
40	12-4-8	)	←	←	←	→	)
41	11	–	←	←	←	→	–
42	12	+	←	←	←	→	+
43	12-6-8	<	←	←	←	→	<
44	3-8	=	←	←	←	→	=
45	6-8	>	←	←	←	→	>
46	2-8	&	–	–	–	–	–
47	11-3-8	$	←	←	←	→	$
50	11-4-8	*	←	←	←	→	*
51	0-4-8	(	←	←	←	→	(
52	0-5-8	%	"	"	"	"	"
53	5-8	:	←	←	←	→	:
54	12-0	?	←	←	←	→	?
55	11-0	!	←	←	←	→	!
56	0-3-8	, (comma)	←	←	←	→	, (comma)
57	0-6-8	\	◇	◇	◇	⬡ S	◇ S
60	0	0	0	0	Ø	0	Ø
61	1	1	←	←	←	→	1
62	2	2	←	←	←	→	2
63	3	3	←	←	←	→	3
64	4	4	←	←	←	→	4
65	5	5	←	←	←	→	5
66	6	6	←	←	←	→	6
67	7	7	←	←	←	→	7
70	8	8	←	←	←	→	8
71	9	9	←	←	←	→	9
72	4-8	'(apostrophe)	←	←	←	→	'(apostrophe)
73	11-6-8	;	←	←	←	→	;
74	0-1	/	←	←	←	→	/
75	12-3-8	.(period)	←	←	←	→	.(period)
76	0-7-8	¤	▱	▱	▱	spec	¤
77	0-2-8	≠ or stop	↑	↑	↑	↑	↑

Note 1: When either 00_8 or 04_8 code is received for either the CRT or the PAGEWRITER, it is completely ignored.

Note 2: When the 03_8 code is received for the CRT, the CRT addressing circuits are set to display the symbol for the next character code as the first symbol on the next line.

Note 3: When the 03_8 code is received for the PAGEWRITER, a combination carriage return and line feed action is initiated at the PAGEWRITER.

APPENDIX D. PROCESSOR UNIT DIFFERENCES

INSTRUCTIONS OR CAPABILITY	TYPE 3011-95	TYPE 3011-97	TYPE 3011-99
Prevent All I/O Interrupts and Jump (PAIJ-72,13)	Privileged	Privileged	Not Privileged
Store Channel Number (SCN-72,14):			
For a = 0	As defined	As defined	As defined
For a = 1	As defined	Undefined for some; as defined for others	Undefined
Initiate Interprocessor Interrupt (III-73,14), for both a = 0 and a = 1	As defined	Undefined for some; as defined for others	Undefined
Enable Day Clock (EDC-73,14), for a = 11_8	As defined	Undefined for some; as defined for others	Undefined
Disable Day Clock (DDC-73,14), for a = 12_8	As defined	Undefined for some; as defined for others	Undefined
Load Last Address Register (LLA-73,16), for a=1	As defined	Undefined for some; as defined for others	Undefined
Test and Set (TS-73,17)	As defined	As defined	Illegal instruction fault interrupt
Maximum Main Storage Size	262K	262K	131K
Interpretation of bit 17 or PSR	Quarter word indicator	Quarter word indicator	Ignored
Is ESI quarter word logic included?	Yes	Yes	No
Used with Availability Control Unit	Yes	No	No
Effect of the CPU in real time mode on Halt Jump (HJ) and Halt Keys And Jump (HKJ)	No effect	Prevents halt for some; no effect for others	Prevents halt
Time needed to update real time clock	0.375 μs	0.375 μs	0.625 μs
CPU operability if a main storage power supply cabinet is inoperative:			
Noninterleaved main storage	Not applicable	Operable	Operable
Interleaved main storage	Operable	Operable for some CPU's; nonoperable for others	Not applicable
Is subsystem operable if power is down in a CPU attached to the SPI?	Yes	Yes, if CPU includes shared subsystem kit	No
Fixed Address:			
Status word made available with external interrupt	200_8 + CPU#	200_8 + CPU#	230_8
Test and set interrupt entrance	244_8 + MSR	244_8 + MSR	No Test and Set instruction

APPENDIX E. INSTRUCTION REPERTOIRE

E.1. INTRODUCTION

Table E-1 lists the UNIVAC 1108 instruction repertoire in function code order. This table also provides the mnemonic, the instruction name, a brief description, and the execution time for each instruction. Table E-2 cross-references the mnemonic with the function codes.

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION (2)	EXECUTION TIME IN μSEC (1)
f	j				
00	–	–	Illegal Code	Causes illegal instruction interrupt to address 241_8	–
01	0-15	S,SA	Store A	(A) → U	.75
02	0-15	SN, SNA	Store Negative A	-(A) → U	.75
03	0-15	SM, SMA	Store Magnitude A	|(A)| → U	.75
04	0-15	S,SR	Store R	(R_a) → U	.75
05	0-15	SZ	Store Zero	Zeros → U	.75
06	0-15	S, SX	Store X	(X_a) → U	.75
07	–	–	Illegal Code	Causes illegal instruction interrupt to address 241_8	–
10	0-17	L, LA	Load A	(U) → A	.75
11	0-17	LN,LNA	Load Negative A	-(U) → A	.75
12	0-17	LM, LMA	Load Magnitude A	|(U)| → A	.75
13	0-17	LNMA	Load Negative Magnitude A	-|(U)| → A	.75
14	0-17	A, AA	Add to A	(A) + (U) → A	.75
15	0-17	AN, ANA	Add Negative to A	(A) - (U) → A	.75
16	0-17	AM, AMA	Add Magnitude to A	(A) + |(U)| → A	.75
17	0-17	ANM, ANMA	Add Negative Magnitude to A	(A) - |(U)| → A	.75
20	0-17	AU	Add Upper	(A) + (U) → A + 1	.75
21	0-17	ANU	Add Negative Upper	(A) - (U) → A + 1	.75
22	0-15	BT	Block Transfer	$(X_x + u) \rightarrow X_a + u$; repeat K times	2.25 + 1.5K always

See notes at end of table.

Table E-1. Instruction Repertoire (Part 1 of 10)

FUNCTION CODE (OCTAL) f	FUNCTION CODE (OCTAL) j	MNEMONIC	INSTRUCTION	DESCRIPTION②	EXECUTION TIME IN μSEC ①
23	0-17	L, LR	Load R	$(U) \rightarrow R_a$	.75
24	0-17	A, AX	Add to X	$(X_a) + (U) \rightarrow X_a$	.75
25	0-17	AN, ANX	Add Negative to X	$(X_a) - (U) \rightarrow X_a$	.75
26	0-17	LXM	Load X Modifier	$(U) \rightarrow X_{a_{17-0}}$; $X_{a_{35-18}}$ unchanged	.875
27	0-17	L, LX	Load X	$(U) \rightarrow X_a$	.75
30	0-17	MI	Multiply Integer	$(A) \cdot (U) \rightarrow A, A+1$	2.375
31	0-17	MSI	Multiply Single Integer	$(A) \cdot (U) \rightarrow A$	2.375
32	0-17	MF	Multiply Fractional	$(A) \cdot (U) \rightarrow A, A+1$	2.375
33	–	–	Illegal Code	Causes illegal instruction interrupt to address 241_8	–
34	0-17	DI	Divide Integer	$(A, A+1) \div (U) \rightarrow A$; Remainder $\rightarrow A+1$	10.125
35	0-17	DSF	Divide Single Fractional	$(A) \div (U) \rightarrow A+1$	10.125
36	0-17	DF	Divide Fractional	$(A, A+1) \div (U) \rightarrow A$; Remainder $\rightarrow A+1$	10.125
37	–	–	Illegal Code	Causes illegal instruction interrupt to address 241_8	–
40	0-17	OR	Logical OR	(A) OR (U) $\rightarrow A+1$	.75
41	0-17	XOR	Logical Exclusive OR	(A) XOR (U) $\rightarrow A+1$	.75
42	0-17	AND	Logical AND	(A) AND (U) $\rightarrow A+1$	.75
43	0-17	MLU	Masked Load Upper	[(U) AND (R2)] OR [(A) AND (R2)′] $\rightarrow A+1$	.75
44	0-17	TEP	Test Even Parity	Skip NI if (U) AND (A) have even parity	2.00 skip 1.25 NI
45	0-17	TOP	Test Odd Parity	Skip NI if (U) AND (A) have odd parity	2.00 skip 1.25 NI
46	0-17	LXI	Load X Increment	$(U) \rightarrow X_{a_{35-18}}$; $X_{a_{17-0}}$ unchanged	1.00
47	0-17	TLEM	Test Less Than or Equal to Modifier	Skip NI if $(U) \leq (X_a)_{17-0}$; always $(X_a)_{17-0} + (X_a)_{35-18} \rightarrow X_{a_{17-0}}$	1.75 skip 1.00 NI
		TNGM	Test Not Greater Than Modifier		
50	0-17	TZ	Test Zero	Skip NI if $(U) = \pm 0$	1.625 skip .875 NI

See notes at end of table.

Table E-1. Instruction Repertoire
(Part 2 of 10)

FUNCTION CODE (OCTAL) f	FUNCTION CODE (OCTAL) j	MNEMONIC	INSTRUCTION	DESCRIPTION②	EXECUTION TIME IN μSEC ①
51	0-17	TNZ	Test Nonzero	Skip NI if (U) ≠ ±0	1.625 skip .875 NI
52	0-17	TE	Test Equal	Skip NI if (U) = (A)	1.625 skip .875 NI
53	0-17	TNE	Test Not Equal	Skip NI if (U) ≠ (A)	1.625 skip .875 NI
54	0-17	TLE TNG	Test Less Than or Equal Test Not Greater	Skip NI if (U) ≤ (A)	1.625 skip .875 NI
55	0-17	TG	Test Greater	Skip NI if (U) > (A)	1.625 skip .875 NI
56	0-17	TW	Test Within Range	Skip NI if (A) < (U) ≤ (A + 1)	1.75 skip 1.00 NI
57	0-17	TNW	Test Not Within Range	Skip NI if (U) ≤ (A) or (U) > (A + 1)	1.75 skip 1.00 NI
60	0-17	TP	Test Positive	Skip NI if $(U)_{35} = 0$	1.50 skip .75 NI
61	0-17	TN	Test Negative	Skip NI if $(U)_{35} = 1$	1.50 skip .75 NI
62	0-17	SE	Search Equal	Skip NI if (U) = (A), else repeat	2.25 + .75K always
63	0-17	SNE	Search Not Equal	Skip NI if (U) ≠ (A), else repeat	2.25 + .75K always
64	0-17	SLE SNG	Search Less Than or Equal Search Not Greater	Skip NI if (U) ≤ (A), else repeat	2.25 + .75K always
65	0-17	SG	Search Greater	Skip NI if (U) > (A), else repeat	2.25 + .75K always
66	0-17	SW	Search Within Range	Skip NI if (A) < (U) ≤ (A + 1), else repeat	2.25 + .75K always
67	0-17	SNW	Search Not Within Range	Skip NI if (U) ≤ (A) or (U) > (A + 1), else repeat	2.25 + .75K always
70	③	JGD	Jump Greater and Decrement	Jump to U if $(\text{Control Register})_{ja} > 0$; go to NI if $(\text{Control Register})_{ja} \leq 0$; always $(\text{Control Register})_{ja}\text{-}1 \rightarrow \text{Control Register}_{ja}$	1.50 jump .75 NI
71	00	MSE	Mask Search Equal	Skip NI if (U) AND (R2) = (A) AND (R2), else repeat	2.25 + .75K always

See notes at end of table.

Table E-1. Instruction Repertoire (Part 3 of 10)

FUNCTION CODE (OCTAL) f	FUNCTION CODE (OCTAL) j	MNEMONIC	INSTRUCTION	DESCRIPTION②	EXECUTION TIME IN μSEC ①
71	01	MSNE	Mask Search Not Equal	Skip NI if (U) AND (R2) ≠ (A) AND (R2), else repeat	2.25 + .75K always
71	02	MSLE	Mask Search Less Than or Equal	Skip NI if (U) AND (R2) ≤ (A) AND (R2), else repeat	2.25 + .75K always
		MSNG	Mask Search Not Greater		
71	03	MSG	Mask Search Greater	Skip NI if (U) AND (R2) > (A) AND (R2), else repeat	2.25 + .75K always
71	04	MSW	Masked Search Within Range	Skip NI if (A) AND (R2) < (U) AND (R2) ≤ (A + 1) AND (R2), else repeat	2.25 + .75K always
71	05	MSNW	Masked Search Not Within Range	Skip NI if (U) AND (R2) ≤ (A) AND (R2) or (U) AND (R2) > (A + 1) AND (R2), else repeat	2.25 + .75K always
71	06	MASL	Masked Alphanumeric Search Less Than or Greater	Skip NI if (U) AND (R2) ≤ (A) AND (R2), else repeat	2.25 + .75K always
71	07	MASG	Masked Alphanumeric Search Greater	Skip NI if (U) AND (R2) > (A) AND (R2), else repeat	2.25 + .75K always
71	10	DA	Double Precision Fixed-Point Add	(A,A + 1) + (U,U + 1) → A,A + 1	1.625
71	11	DAN	Double Precision Fixed-Point Add Negative	(A,A + 1) - (U,U + 1) → A,A + 1	1.625
71	12	DS	Double Store A	(A,A + 1) → U,U + 1	1.50
71	13	DL	Double Load A	(U,U + 1) → A,A + 1	1.50
71	14	DLN	Double Load Negative A	-(U,U + 1) → A,A + 1	1.50
71	15	DLM	Double Load Magnitude A	|(U,U + 1)| → A,A + 1	1.50
71	16	DJZ	Double Precision Jump Zero	Jump to U if (A,A + 1) = ± 0; go to NI if (A,A + 1) ≠ ± 0	1.625 jump .875 NI
71	17	DTE	Double Precision Test Equal	Skip NI if (U,U + 1) = (A,A+1)	2.375 skip 1.625 NI
72	00	–	Illegal Code	Causes illegal instruction interrupt to address 241_8	–
72	01	SLJ	Store Location and Jump	(P) – Base Address Modifier [BI or BD] → U_{17-0}; jump to U + 1	2.125 always
72	02	JPS	Jump Positive and Shift	Jump to U if $(A)_{35} = 0$; go to NI if $(A)_{35} = 1$; always shift (A) left circularly one bit position	1.50 jump .75 NI always

See notes at end of table.

Table E-1. Instruction Repertoire
(Part 4 of 10)

FUNCTION CODE (OCTAL)		MNEMONIC	INSTRUCTION	DESCRIPTION②	EXECUTION TIME IN μSEC ①
f	j				
72	03	JNS	Jump Negative and Shift	Jump to U if $(A)_{35} = 1$; go to NI if $(A)_{35} = 0$; always shift (A) left circularly on one bit position	1.50 jump .75 NI always
72	04	AH	Add Halves	$(A)_{35-18} + (U)_{35-18} \rightarrow A_{35-18}$; $(A)_{17-0} + (U)_{17-0} \rightarrow A_{17-0}$	.75 always
72	05	ANH	Add Negative Halves	$(A)_{35-18} - (U)_{35-18} \rightarrow A_{35-18}$; $(A)_{17-0} - (U)_{17-0} \rightarrow A_{17-0}$	.75 always
72	06	AT	Add Thirds	$(A)_{35-24} + (U)_{35-24} \rightarrow A_{35-24}$; $(A)_{23-12} + (U)_{23-12} \rightarrow A_{23-12}$; $(A)_{11-0} + (U)_{11-0} \rightarrow A_{11-0}$	.75 always
72	07	ANT	Add Negative Thirds	$(A)_{35-24} - (U)_{35-24} \rightarrow A_{35-24}$; $(A)_{23-12} - (U)_{23-12} \rightarrow A_{23-12}$; $(A)_{11-0} - (U)_{11-0} \rightarrow A_{11-0}$	.75 always
72	10	EX	Execute	Execute the instruction at U	.75 always
72	11	ER	Executive Return	Causes executive return interrupt to address 242_8	1.375 always
72	12	–	Illegal Code	Causes illegal instruction interrupt to address 241_8	–
72	13	PAIJ	Prevent All I/O Interrupts and Jump	Prevent all I/O interrupts and jump to U	.75 always
72	14	SCN	Store Channel Number	If a = 0: channel number → U_{3-0}; if a = 1: channel number → U_{3-0} and CPU number → U_{5-4}	.75
72	15	LPS	Load Processor State Register	(U) → Processor State Register	.75
72	16	LSL	Load Storage Limits Register	(U) → SLR	.75
72	17	–	Illegal Code	Causes illegal instruction interrupt to address 241_8	–
73	00	SSC	Single Shift Circular	Shift (A) right circularly U-places	.75 always
73	01	DSC	Double Shift Circular	Shift (A,A + 1) right circularly U-places	.875 always
73	02	SSL	Single Shift Logical	Shift (A) right U-places; zerofill	.75 always
73	03	DSL	Double Shift Logical	Shift (A,A + 1) right U-places; zerofill	.875 always
73	04	SSA	Single Shift Algebraic	Shift (A) right U-places; signfill	.75 always
73	05	DSA	Double Shift Algebraic	Shift (A,A + 1) right U-places; signfill	.875 always

See notes at end of table.

Table E-1. Instruction Repertoire
(Part 5 of 10)

FUNCTION CODE (OCTAL) f	FUNCTION CODE (OCTAL) j	MNEMONIC	INSTRUCTION	DESCRIPTION②	EXECUTION TIME IN μSEC ①
73	06	LSC	Load Shift and Count	(U) → A, shift (A) left circularly until $(A)_{35} \neq (A)_{34}$; NUMBER OF SHIFTS → A + 1	1.125
73	07	DLSC	Double Load Shift and Count	(U,U + 1) → A,A + 1; shift (A,A + 1) left circularly until $(A,A+1)_{71} \neq (A,A+1)_{70}$; NUMBER OF SHIFTS → A + 2	2.125
73	10	LSSC	Left Single Shift Circular	Shift (A) left circularly U-places	.75 always
73	11	LDSC	Left Double Shift Circular	Shift (A,A + 1) left circularly U-places	.875 always
73	12	LSSL	Left Single Shift Logical	Shift (A) left U-places; zerofill	.75 always
73	13	LDSL	Left Double Shift Logical	Shift (A,A + 1) left U-places; zerofill	.875 always
73	14	III (a = 0 or 1)	Initiate Interprocessor Interrupt	Initiate interprocessor interrupt	.75 always
		ALRM ($a = 10_8$)	Alarm	Turn on alarm	.75 always
		EDC ($a = 11_8$)	Enable Day Clock	Enable day clock	.75 always
		DDC ($a = 12_8$)	Disable Day Clock	Disable day clock	.75 always
73	15	SIL	Select Interrupt Locations	(a) → MSR	.75 always
73	16	LCR (a = 0)	Load Channel Select Register	$(U)_{3-0}$ → CSR	.875
		LLA (a = 1)	Load Last Address Register	$(U)_{2-0}$ → LAR	.875
73	17	TS	Test and Set	If $(U)_{30} = 1$, interrupt to address 244_8; if $(U)_{30} = 0$, go to NI; always $01_8 \rightarrow U_{35-30}$; $(U)_{29-0}$ unchanged	Alternate bank: 1.625 interrupt .875 NI Same bank: 2.0 interrupt 2.0 NI
74	00	JZ	Jump Zero	Jump to U if (A) = ± 0; go to NI if (A) ≠ ± 0	1.50 jump .75 NI always
74	01	JNZ	Jump Nonzero	Jump to U if (A) ≠ ± 0; go to NI if (A) = ± 0	1.50 jump .75 NI always

See notes at end of table.

Table E-1. Instruction Repertoire (Part 6 of 10)

FUNCTION CODE (OCTAL) f	FUNCTION CODE (OCTAL) j	MNEMONIC	INSTRUCTION	DESCRIPTION②	EXECUTION TIME IN μSEC ①
74	02	JP	Jump Positive	Jump to U if $(A)_{35} = 0$; go to NI if $(A)_{35} = 1$	1.50 jump .75 NI always
74	03	JN	Jump Negative	Jump to U if $(A)_{35} = 1$; go to NI if $(A)_{35} = 0$	1.50 jump .75 NI always
74	04	JK J	Jump Keys Jump	Jump to U if a = 0 or if a = lit select jump indicator; go to NI if neither is true	.75 always
74	05	HKJ HJ	Halt Keys and Jump Halt Jump	Stop if a = 0 or if [a AND lit select stop indicators] ≠ 0; on restart or continuation, jump to U	.75 always
74	06	NOP	No operation	Proceed to next instruction	.75 always
74	07	AAIJ	Allow All I/O Interrupts and Jump	Allow all I/O interrupts and jump to U	.75 always
74	10	JNB	Jump No Low Bit	Jump to U if $(A)_0 = 0$; go to NI if $(A)_0 = 1$	1.50 jump .75 NI always
74	11	JB	Jump Low Bit	Jump to U if $(A)_0 = 1$; go to NI if $(A)_0 = 0$	1.50 jump .75 NI always
74	12	JMGI	Jump Modifier Greater and Increment	Jump to U if $(X_a)_{17-0} > 0$; go to NI if $(X_a)_{17-0} \leq 0$; always $(X_a)_{17-0} + (X_a)_{35-18} \rightarrow X_{a_{17-0}}$	1.50 jump .75 NI always
74	13	LMJ	Load Modifier and Jump	(P) − BASE ADDRESS MODIFIER [BI or BD] $\rightarrow X_{a_{17-0}}$; jump to U	.875 always
74	14	JO	Jump Overflow	Jump to U if D1 of PSR = 1; go to NI if D1 = 0	1.50 jump .75 NI always
74	15	JNO	Jump No Overflow	Jump to U if D1 of PSR = 0; go to NI if D1 = 1	1.50 jump .75 NI always
74	16	JC	Jump Carry	Jump to U if D0 of PSR = 1; go to NI if D0 = 0	1.50 jump .75 NI always
74	17	JNC	Jump No Carry	Jump to U if D0 of PSR = 0; go to NI if D0 = 1	1.50 jump .75 NI always

See notes at end of table.

Table E-1. Instruction Repertoire
(Part 7 of 10)

FUNCTION CODE (OCTAL) f	FUNCTION CODE (OCTAL) j	MNEMONIC	INSTRUCTION	DESCRIPTION②	EXECUTION TIME IN μSEC ①
75	00	LIC	Load Input Channel	For channel [a OR CSR]: (U) → IACR; set input active; clear input monitor	.75
75	01	LICM	Load Input Channel and Monitor	For channel [a OR CSR]: (U) → IACR; set input active; set input monitor	.75
75	02	JIC	Jump On Input Channel Busy	Jump to U if input active is set for channel [a OR CSR]; go to NI if input active is clear	.75 always
75	03	DIC	Disconnect Input Channel	For channel [a OR CSR]: clear input active; clear input monitor	.75 always
75	04	LOC	Load Output Channel	For channel [a OR CSR]: (U) → OACR; set output active; clear output monitor; clear external monitor (ISI only)	.75
75	05	LOCM	Load Output Channel and Monitor	For channel [a OR CSR]: (U) → OACR; set output active; set output monitor; clear external function (ISI only)	.75
75	06	JOC	Jump on Output Channel Busy	Jump to U if output active is set for channel [a OR CSR]; go to NI if output active is clear	.75 always
75	07	DOC	Disconnect Output Channel	For channel [a OR CSR]: clear output active; clear output monitor; clear external function	.75 always
75	10	LFC	Load Function in Channel	For channel [a OR CSR]: (U) → OACR; set output active (ISI only), external function, and force external function; clear output monitor (ISI only)	.75
75	11	LFCM	Load Function in Channel and Monitor	For channel [a OR CSR]: (U) → OACR; set output active (ISI only), external function, force external function, and output monitor (ISI only)	.75
75	12	JFC	Jump On Function in Channel	Jump to U if force external function is set for channel [a OR CSR]; go to NI if force external function is clear	.75 always

See notes at end of table.

Table E-1. Instruction Repertoire
(Part 8 of 10)

FUNCTION CODE (OCTAL) f	FUNCTION CODE (OCTAL) j	MNEMONIC	INSTRUCTION	DESCRIPTION②	EXECUTION TIME IN μSEC ①
75	13	–	Illegal Code	If guard mode is set, causes guard mode interrupt to address 243_8. If guard mode is not set, same as NOP	.75 always
75	14	AACI	Allow All Channel External Interrupts	Allow all external interrupts	.75 always
75	15	PACI	Prevent All Channel External Interrupts	Prevent all external interrupts	.75 always
75	16	–	Illegal Code	If guard mode is set, causes guard mode interrupt to address 243_8. If guard mode is not set, same as NOP	.75 always
75	17	–	Illegal Code		
76	00	FA	Floating Add	(A) + (U) → A; RESIDUE → A+ 1	1.875
76	01	FAN	Floating Add Negative	(A) - (U) → A; RESIDUE → A+ 1	1.875
76	02	FM	Floating Multiply	(A) · (U) → A,A+ 1	2.625
76	03	FD	Floating Divide	(A) ÷ (U) → A; REMAINDER → A + 1	8.25④
76	04	LUF	Load and Unpack Floating	$\|(U)\|_{34-27} \rightarrow A_{7-0}$, zerofill; $(U)_{26-0} \rightarrow A + 1_{26-0}$, signfill	.75 always
76	05	LCF	Load and Convert to Floating	$(U)_{35} \rightarrow A + 1_{35}$; [NORMALIZED $(U)]_{26-0} \rightarrow A + 1_{26-0}$; if $(U)_{35} = 0$, $(A)_{7-0} \pm$ NORMALIZING COUNT $\rightarrow A + 1_{34-27}$; if $(U)_{35} = 1$, ones complement of [$(A)_{7-0} \pm$ NORMALIZING COUNT] $\rightarrow A + 1_{34-27}$	1.125
76	06	MCDU	Magnitude of Characteristic Difference to Upper	$\|\|(A)\|_{35-27} - \|(U)\|_{35-27}\| \rightarrow A + 1_{8-0}$; ZEROS → $A + 1_{35-9}$	.75 always
76	07	CDU	Characteristic Difference to Upper	$\|(A)\|_{35-27} - \|(U)\|_{35-27} \rightarrow A + 1_{8-0}$; SIGN BITS → $A + 1_{35-9}$	.75 always
76	10	DFA	Double Precision Floating Add	(A,A + 1) + (U,U+ 1) → A,A+ 1	2.625
76	11	DFAN	Double Precision Floating Add Negative	(A,A + 1) - (U,U+ 1) → A,A+ 1	2.625
76	12	DFM	Double Precision Floating Multiply	(A,A + 1) · (U,U+ 1) → A,A+ 1	4.25
76	13	DFD	Double Precision Floating Divide	(A,A + 1) ÷ (U,U+ 1) → A,A+ 1	17.25⑤

See notes at end of table.

Table E-1. Instruction Repertoire (Part 9 of 10)

FUNCTION CODE (OCTAL) f	FUNCTION CODE (OCTAL) j	MNEMONIC	INSTRUCTION	DESCRIPTION (2)	EXECUTION TIME IN μSEC (1)
76	14	DFU	Double Load and Unpack Floating	$\|(U)\|_{34-24} \rightarrow A_{10-0}$, zerofill; $(U)_{23-0} \rightarrow A+1_{23-0}$, signfill; $(U+1) \rightarrow A+2$	1.50
76	15	DFP	Double Load and Convert to Floating	$(U) \rightarrow A+1_{35}$; $[\text{NORMALIZED } (U,U+1)]_{59-0} \rightarrow A+1_{23-0}$ and $A+2$; if $(U)_{35}=0$, $(A)_{10-0} \pm$ NORMALIZING COUNT $\rightarrow A+1_{34-24}$; if $(U)_{35}=1$, ones complement of $[(A)_{10-0} \pm$ NORMALIZING COUNT$] \rightarrow A+1_{34-24}$	
76	16	FEL	Floating Expand and Load	If $(U)_{35}=0$, $(U)_{35-27}+1600_8 \rightarrow A_{35-24}$; if $(U)_{35}=1$, $(U)_{35-27} - 1600_8 \rightarrow A_{35-24}$; $(U)_{26-3} \rightarrow A_{23-0}$; $(U)_{2-0} \rightarrow A+1_{35-33}$; $(U)_{35} \rightarrow A+1_{32-0}$	1.00
76	17	FCL	Floating Compress and Load	If $(U)_{35}=0$, $(U)_{35-24} -1600_8 \rightarrow A_{35-27}$; if $(U)_{35}=1$, $(U)_{35-24} + 1600_8 \rightarrow A_{35-27}$; $(U)_{23-0} \rightarrow A_{26-3}$; $(U+1)_{35-33} \rightarrow A_{2-0}$	1.625
77	0-17	–	Illegal Code	Causes illegal instruction interrupt to address 241_8	–

Notes:

(1) The execution times given are for alternate bank memory access; for same bank memory access, execution time is .75 microsecond greater. Exceptions to this either show the execution times for both types of memory access or include the word "always" to indicate that the execution time is the same regardless of the type of memory access.

For function codes 01 through 06 and 22, add .375 microsecond to the execution times for 6-bit and 12-bit writes.

The execution time for a Block Transfer or any of the search instructions depends on the number of repetitions (K) required; that is, the number of words in the block being transferred or the number of words searched before a find is made.

(2) NI stands for Next Instruction

(3) The a and j fields together serve to specify any of the 128 control registers.

(4) If 28 rather than 27 subtractions are performed, add .25 microsecond to the execution time.

(5) If 61 rather than 60 subtractions are performed, add .25 microsecond to the execution time.

Table E–1. Instruction Repertoire
(Part 10 of 10)

MNEMONIC	FUNCTION CODE (OCTAL) f	j	MNEMONIC	FUNCTION CODE (OCTAL) f	j	MNEMONIC	FUNCTION CODE (OCTAL) f	j	MNEMONIC	FUNCTION CODE (OCTAL) f	j
A	14		DSF	35		LIC	75	00	SIL	73	15
A	24		DSL	73	03	LICM	75	01	SLE	64	
AA	14		DTE	71	17	LLA	73 a=1	16	SLJ	72	01
AACI	75	14	EDC	73 a=11_8	14				SM	03	
AAIJ	74	07				LM	12		SMA	03	
AH	72	04	ER	72	11	LMA	12		SN	02	
ALRM	73 a=10_8	14	EX	72	10	LMJ	74	13	SNA	02	
			FA	76	00	LN	11		SNE	63	
AM	16		FAN	76	01	LNA	11		SNG	64	
AMA	16		FCL	76	17	LNMA	13		SNW	67	
AN	15		FD	76	03	LOC	75	04	SR	04	
AN	25		FEL	76	16	LOCM	75	05	SSA	73	04
ANA	15		FM	76	02	LPS	72	15	SSC	73	00
AND	42		HJ	74	05	LR	23		SSL	73	02
ANH	72	05	HKJ	74	05	LSC	73	06	SW	66	
ANM	17		III	73 a=0 or 1	14	LSL	72	16	SX	06	
ANMA	17					LSSC	73	10	SZ	05	
ANT	72	07	J	74	04	LSSL	73	12	TE	52	
ANU	21		JB	74	11	LUF	76	04	TEP	44	
ANX	25		JC	74	16	LX	27		TG	55	
AT	72	06	JFC	75	12	LXI	46		TLE	54	
AU	20		JGD	70		LXM	26		TLEM	47	
AX	24		JIC	75	02	MASG	71	07	TN	61	
BT	22		JK	74	04	MASL	71	06	TNE	53	
CDU	76	07	JMGI	74	12	MCDU	76	06	TNG	54	
DA	71	10	JN	74	03	MF	32		TNGM	47	
DAN	71	11	JNB	74	10	MI	30		TNW	57	
DDC	73 a=12_8	14	JNC	74	17	MLU	43		TNZ	51	
			JNO	74	15	MSE	71	00	TOP	45	
DF	36		JNS	72	03	MSG	71	03	TP	60	
DFA	76	10	JNZ	74	01	MSI	31		TS	73	17
DFAN	76	11	JO	74	14	MSLE	71	02	TW	56	
DFD	76	13	JOC	75	06	MSNE	71	01	TZ	50	
DFM	76	12	JP	74	02	MSNG	71	02	XOR	41	
DFP	76	15	JPS	72	02	MSNW	71	05	—	00	
DFU	76	14	JZ	74	00	MSW	71	04	—	07	
DI	34		L	10		NOP	74	06	—	33	
DIC	75	03	L	23		OR	40		—	37	
DJZ	71	16	L	27		PACI	75	15	—	72	00
DL	71	13	LA	10		PAIJ	72	13	—	72	12
DLM	71	15	LCF	76	05	S	01		—	72	17
DLN	71	14	LCR	73 a=0	16	S	04		—	75	13
DLSC	73	07				S	06		—	75	16
DOC	75	07	LDSC	73	11	SA	01		—	75	17
DS	71	12	LDSL	73	13	SCN	72	14	—	77	
DSA	73	05	LFC	75	10	SE	62				
DSC	73	01	LFCM	75	11	SG	65				

Table E-2. Mnemonic/Function Code Cross-Reference


UNIVAC
SPERRY RAND