UNIVAC

DATA PROCESSING DIVISION

1108

MULTI-PROCESSOR SYSTEM

PROCESSOR AND STORAGE

REFERENCE MANUAL

UP-4053

This manual is published by the Univac Division of Sperry Rand Corporation in loose leaf format as 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 programming effort. For the current status of the programming, 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 and/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 respective Systems.

PRINTED IN U.S.A.

CONTENTS

ILLUSTRATIONS

TABLES

1. INTRODUCTION

The UNIVAC 1108 System - the logical, program compatible successor to the UNIVAC 1107 Thin Film Memory Computer - is an integration of system-oriented hardware design, imaginative development and software technology. The result is a system that effectively knows no applications boundaries. It is equally capable in real time, scientific or data processing environments, and of adjusting dynamically to any one or a mixture of these environments.

UNIVAC 1108 SYSTEM FEATURES

- Control Registers: 128 integrated-circuit registers.
- Main Storage: 750 nanosecond complete cycle time multiple modules yield an effective 375 nanosecond operation.
- Store Size: 32,768 to 131,072 36-bit words available in 16,384-word increments.
- Parity: internal parity checking on all data and instruction transfers with an automatic error interrupt for a parity failure.
- Real Time Capability: each I/O channel is equipped with Externally Specified Index (ESI), permitting an unlimited number of communication devices to be multiplexed into a single I/O channel.
- Arithmetic Section: single and double precision floating point hardware allowing the representation of more than 18 significant decimal digits with exponentiation greater than 10 to the $\pm$ 300th power.
- Input/Output: up to 16 independent bi-directional input/output channels are available with the capability of an instantaneous data transfer rate of up to 8,000,000 characters per second.
- Program Interrupts: the synchronization of normal input/output activity, response to real time data handling, and immediate response to abnormal machine conditions is fully automatic.
- Instruction Repertoire: the 1108 instruction word format provides an extremely powerful repertoire of more than 900 unique program operations.

- High Speed Auxiliary Storage: the FH-432 Drum transfers information to and from main storage at 1,440,000 characters per second with an average access time of 4.3 milliseconds.

The UNIVAC 1108 System is controlled by a program of instructions contained in its main storage. This program performs the operations specified by the Function Codes on the data furnished to the processor.

Internal operations in the UNIVAC 1108 Processor are performed in a parallel binary mode. The machine language is binary; therefore data, instructions, and control words must be supplied to the computer in binary form. However, for convenience in programming as well as in monitoring internal operations, octal notation is used to represent the binary configuration.

Instruction Word - The fundamental programming tools are the processing and input-output instructions.

This preliminary description of the UNIVAC 1108 System covers the principle programming aspects of the unit-processor. Most of what is described in this book applies also to the multi-processor system. Full details on the multi-processor capability and use of interleaved storage will be covered in a later manual.

2. GLOSSARY AND CONVENTIONS

1. INSTRUCTION FIELDS:

f -field contains function code designator (f)
j -field contains operand qualifier or minor function code (j)
a -field contains AXR-register designator, channel designator, or console keys designator (a)
x -field contains index register designator (x)
h -field contains index register modification designator (h)
i -field contains indirect addressing designator (i)
u -field contains address or operand designator (u)

2. a-Field DESIGNATOR REFERENCES:

K_a = value of "a" designates console key
C_a = value of "a" designates channel number
A_a = value of "a" designates an A-register within set of A-registers
X_a = value of "a" designates an X-register within set of X-registers
R_a = value of "a" designates an R-register within set of R-registers

3. AXR CONTROL REGISTER SETS:

A = A-registers = Accumulators
X = X-registers = Index Registers
R = R-registers = Special Purpose Registers

4. X-REGISTER SUBSCRIPT:

Subscript M = lower half of X-register (Modifier)
Subscript I = upper half of X-register (Increment)

5. ADDRESSES:

U = Program effective address (Relative Address)
S = Main Storage address (Absolute Address)
S_I = Main Storage address in I-Storage Area
S_D = Main Storage address in D-Storage Area

6. REGISTERS:

P = Program Address Register
CR = Control Registers
AR = Address Registers
SLR = Storage Limits Register
PCR = Processor Control Register
ACR = Access Control Register
PSR = Processor State Register

7. PSR D-field DESIGNATORS

D_0 = Carry Designator
D_1 = Overflow Designator
D_2 = Guard Mode Designator
D_3 = Write-Only Storage Protection Designator
D_4 = 1107 Compatibility Designator
D_5 = Double Precision Underflow Designator
D_6 = Control Register Selection Designator
D_7 = Base Register Suppression Designator
D_8 = Floating point zero designator

8. PSR BASE RELATIVE ADDRESSING FIELDS

B_D - field contains Storage Block Number for D-Base
B_S - field contains Program Effective Switch Point
B_I - field contains Storage Block Number for I-Base

9. SLR FIELDS:

I_L - field contains Storage Block Number for Lower Limit of I-Area
I_U - field contains Storage Block Number for Upper Limit of I-Area
D_L - field contains Storage Block Number for Lower Limit of D-Area
D_U - field contains Storage Block Number for Upper Limit of D-Area

10. SPECIAL SYMBOLS:

( ) = contents of specified register or storage address, subscripts indicate bit positions being considered, a prime (') superscript indicates the ones complement.
|( )| = Absolute value or magnitude.
$\longrightarrow$ = direction of data flow or "goes to"
$\odot$ = logical AND function
$\oplus$ = logical OR function
$\overline{\oplus}$ = logical EXCLUSIVE OR

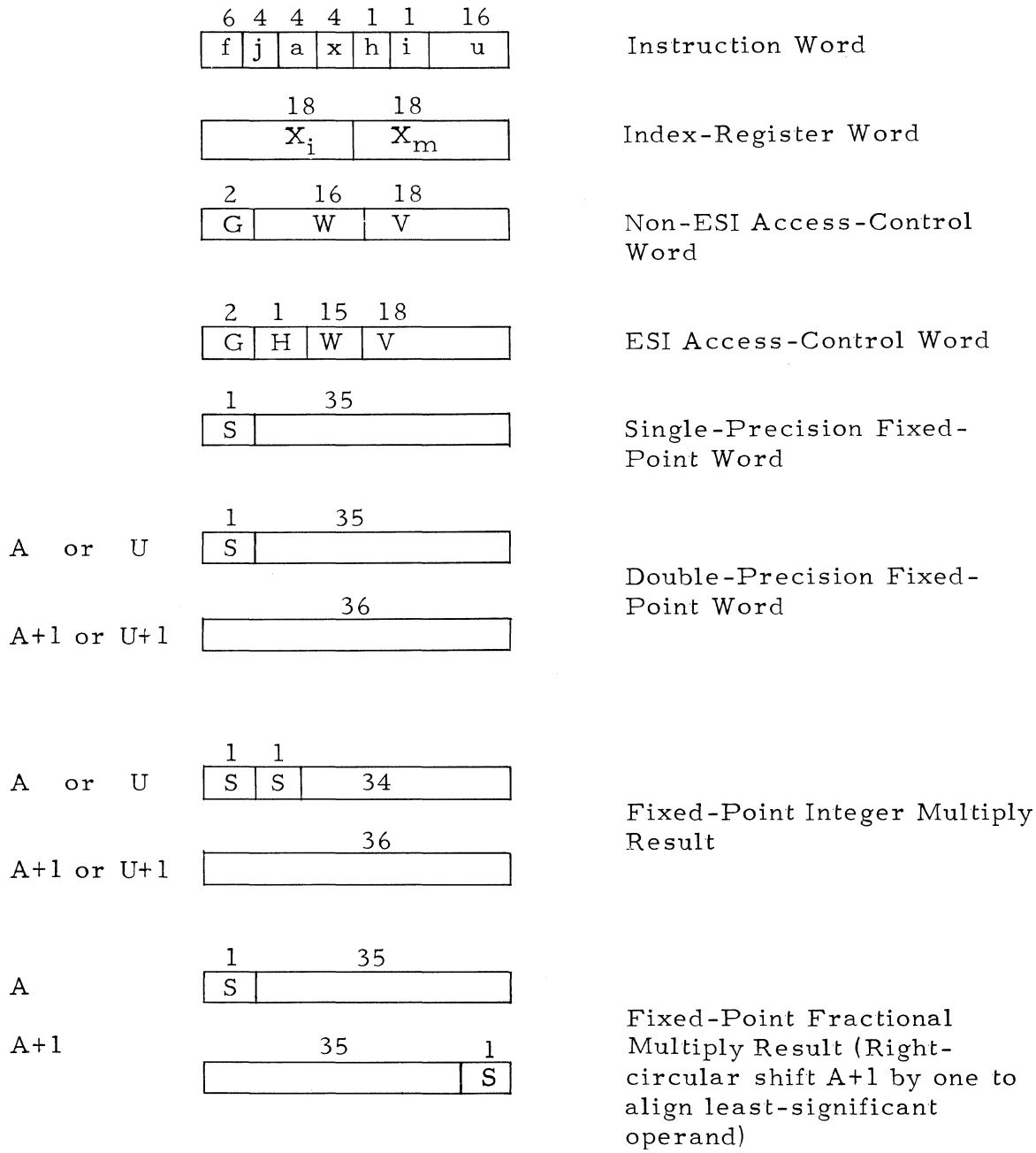
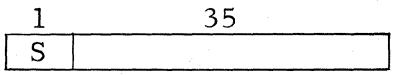
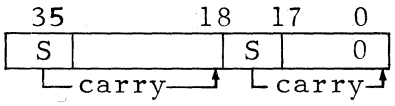


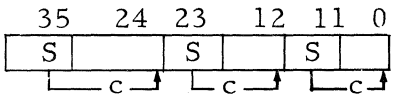
Figure 2-1. UNIVAC 1108 Processor Word Formats
Numbers above segments indicate the number of bits in the segment.



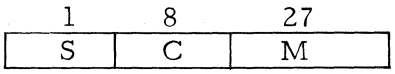
Fixed-Point Multiply Single-Integer Result



Add-Halves Word Format

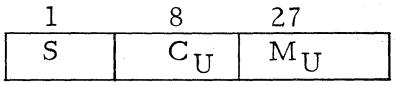


Add-Thirds Word Format

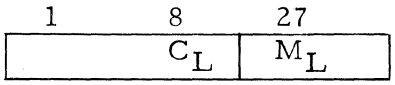


Single-Precision Floating-Point Operand

A



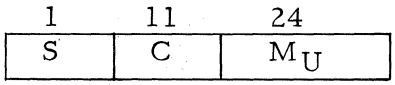
or

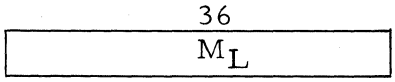


A+1

Single-Precision Floating-Point Result; $C_L = C_U - 27$,

Word 2 contains unnormalized least significant result.





Double-Precision Floating-Point Operand or Result

Figure 2-1 (Cont.) - UNIVAC 1108 Processor Word Formats. Numbers above segments indicate the number of bits in the segment.

3. PROCESSOR INSTRUCTION FORMAT

The instructions are classed according to the fundamental functions they perform. Each class is further divided into sub-classes based on special execution conditions. These conditions normally take the form of indicator settings, specific rules, or special interpretation of the various instruction designators.

When special conditions pertain to more than one instruction, the condition is discussed in the introduction to the pertinent class.

Following the general class and sub-class introduction, each instruction is defined as to its mnemonic code, octal operation code, designators which must be present, a description of and time required for execution, and programming notes where necessary.

The Input/Output instructions are described with respect to the common channel interface the processor presents to I/O subsystems. The introduction to this class states basic requirements and data formats of peripheral subsystems necessary for execution of the I/O operations.

Definitions and Symbology

The basic instruction word is composed of 36-bits grouped into 7 designators as follows.

f	j	a	x	h	i	u
6	4	4	4	1	1	16
35 30	29 26	25 22	21 18	17	16	15 0

f	function code designator
j	operand qualifier or minor function code designator
a	a-register designator
A(a)	A-register designator
X(a)	X-register designator
R(a)	R-register designator
C(a)	I/O channel designator
K(a)	jump key designator
JA(a)	J and A designators combined to form an 8 bit binary designator

x	index register designator
h	index register incrementation designator
i	indirect address designator
u	operand address or operand designator

Function Code Designator, f

The function code, or f designator, contained in the leftmost six bit positions, specifies the particular operation that is to be performed. In instructions where f>70, the j designator becomes part of the function code. Invalid function codes, either f or fj, are fault conditions which cause an error interrupt to occur. In this event, the processor interrupts and transfers control to storage location 000241, a fixed error interrupt location.

Operand Qualifier or Minor Function Code Designator, j

When f<70, the j designator determines whether an entire operand, or only a part of it is to be transferred to or from the arithmetic section. As previously mentioned, in instructions where f>70, j serves as a minor function code rather than as an operand qualifier. When f = 70, the j-designator combines with f to form the function code.

As an operand qualifier in the case of partial word transfers to the arithmetic section, j specifies which half-word, third-word, or sixth-word is to be utilized. The transfer is always to the low order positions of the arithmetic section. In transfers from the arithmetic section, j specifies into which half-word, third-word or sixth-word the low order positions of the word in the arithmetic section will be transferred.

In half-word transfers to the arithmetic section, j can specify whether sign extension is to take place. If it is specified by coding j as 3 or 4, the most significant bit of the half-word fills positions 35 thru 18 of the control register. If sign extension is not specified, i.e., j is coded as 1 or 2, positions 35 through 18 are zero filled.

Sign extension always occurs for third-words and never occurs for sixth-words. No sign extension occurs for transfers from the control registers.

When j equals 16 or 17, the u-field of the instruction becomes the effective operand rather than the address of the operand. When j is coded as 17, sign extension is effective. These two j-field values are not effective when a transfer from the control registers is specified; in such a case, no transfer of data will be made.

When the effective address designates a control register the j designator is not effective; only full words will be transferred.

A-Register Designator, A(a)

The a-designator normally specifies a control register location. For arithmetic operations and some other operations which do not specifically reference other registers, the a-designator specifies one of the 16 A-Registers.

X-Register Designator, X(a)

The a-designator is also used to reference any one of 15 index registers in control memory. An X-Register is implied by the function code in certain instructions. Control register 000000 cannot be normally referenced by an a-designator.

R-Register Designator, R(a)

The a-designator is used to reference any one of 16 R-registers. An R-register is implied by the function code in certain instructions.

C-Register Designator, C(a)

In Input/Output instructions, the a-designator specifies the Input/Output channel affected by the operation, and thereby indirectly defines the appropriate Input/Output Access-Control Word location.

K Jump Key Designator, K(a)

In the Jump-on-Keys instruction, the a-designator specifies one of 15 selective jump switches on the operator's console.

K Stop Key Designator, K(a)

In the Halt and Jump instruction, the a-designator specifies one of 4 selective stop keys on the operator's console.

JA Control Register Designator, JA(a)

In the index jump instruction the j-designator is combined with the a-designator to form an 8 bit designator which may specify any of the 128 words in control registers.

Index Register Designator, x

The format of the indexing information stored at the control register address specified by the x-designator is shown below. Bits 17-00 X_M contain the address modifier which is added to the u address; bits 35-18 (X_I) contain an increment which may, if desired, be used to change the value of X_M. This increment may be positive or negative.

35	X_I	18	17	X_M	00

Index Register Incrementation Designator, h

When the h-designator is coded as 1, the value of X_M in index register X is increased by the value of X_I. This incrementation takes place during the instruction; after the addition of u and the index register, in forming the effective address. When h is 0, no incrementation takes place.

Indirect Addressing Designator, i

The i-designator specifies either direct or indirect addressing of the operand. If i is coded as 0, direct addressing is specified, and u is the effective address of the operand. If i is coded as 1, indirect addressing is specified. Bits 21-00 of the u-addressed operand replace bits 21-00 in the current instruction. Since the 22 bits include the x, h, i, and u-designators, all indexing, index register incrementation, and indirect addressing operations can be cascaded until the i-designator in one of the temporarily formed instructions is 0. If $j<16$, normal partial-word operations on the contents of the address specified by u are performed at the end of cascading. If $j = 16$ or 17, cascading is halted when either the i-designator or the x-designator, or both, become zero; the value in u_{17-00} becomes the actual operand. Thus, for $j = 16$ or 17, indirect addressing is not only conditioned by the i-designator, but is also conditioned upon the x-designator being a non-zero value.

Operand Address or Operand Designator, u

The function of the u-designator depends on the instruction with which it is used. The basic use is as the address of an operand. When modified by indexing, $u+X_M$ form U, the effective address. It is also used as

an actual operand when j is 16 or 17,
a shift-count designator,
an instruction address to which program control can be transferred,
the address of an address when indirect addressing is specified.

4. PROCESSOR AND STORAGE CONTROL

The control section of the 1108 Processor governs the entire system, providing control for the proper sequential execution of the computer program.

Since the control section of the 1108 is complex specific registers will not be described in detail; but rather will be grouped together in theoretical subsections (see Figure 4-1) and only the functions will be described.

The control section contains four subsections, (1) the Program Address Subsection (P), (2) the Program Control Subsection (PC), (3) the Storage Class Control Subsection SCC), and (4) the Index Subsection, all of which are discussed in detail later in this section. The control section also includes the circuits which supply the control signals necessary to synchronize the execution of instructions.

Method of Operation

The address of the Instruction Word to be executed is stored in the P-Subsection. The contents of the P-Subsection is increased by 1 each time an Instruction word is processed. This incrementation is accomplished automatically by the Index Adder which is in the Index Subsection. Each Instruction Word is then transferred in successive order to the PC Subsection for decoding and translation. This translation determines the computer operation to be performed.

In most Instruction Words, a u-Field references an address in either main storage of the Control registers. When the u-field and the Next Instruction word (NI) are in the same main storage module, the NI cannot be addressed until the current one has been fully executed. If they are in different main storage modules, the NI can be addressed before the present instruction has been fully operated upon.

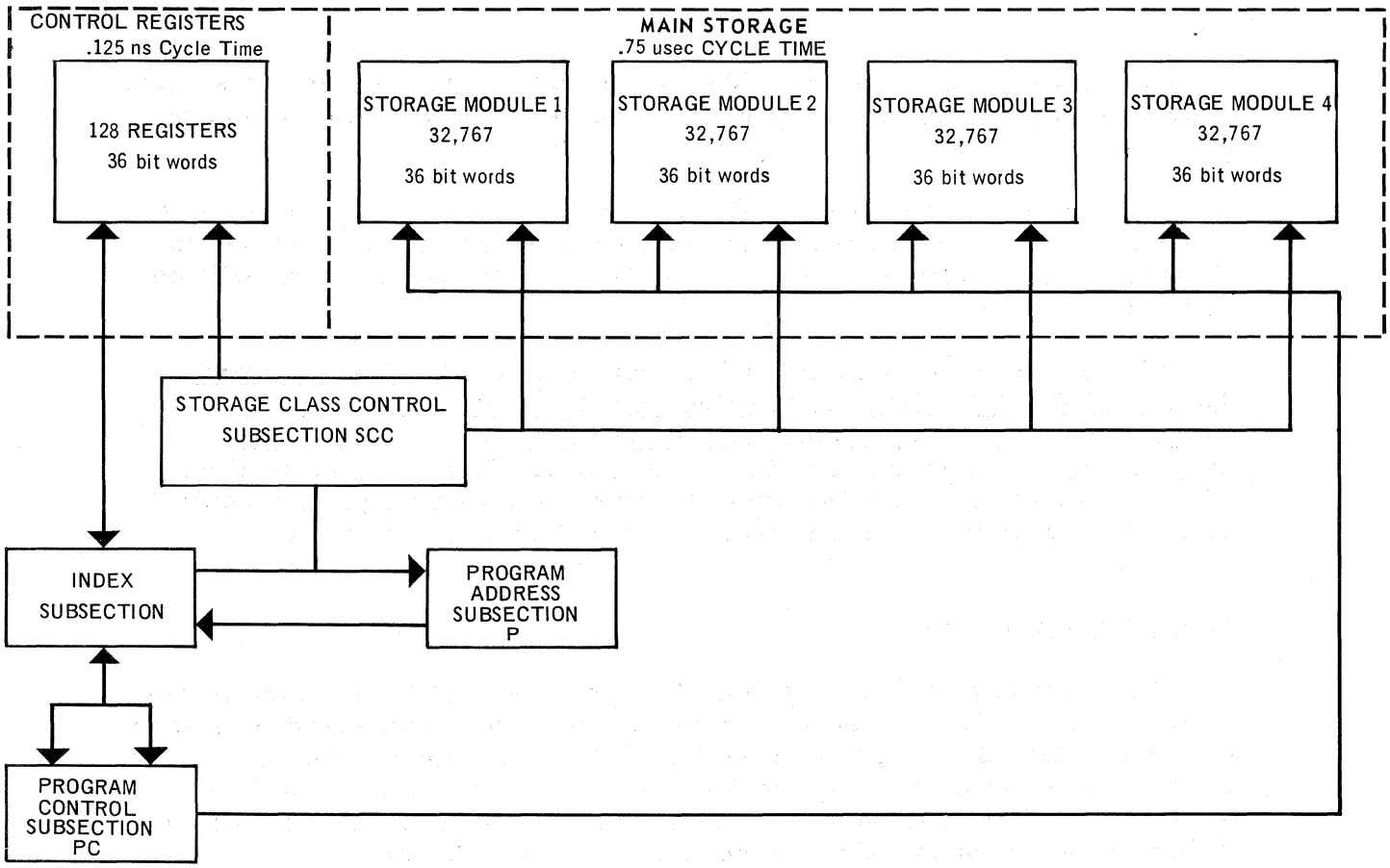


Figure 4–1. UNIVAC 1108 Processor Storage and Control Section.

The u-field may or may not be modified to form the effective operand address, U, depending upon the Instruction Word. All transfers from the PC Subsection to storage addressing circuitry are made through the Index Subsection; the necessary address modification is accomplished at this point.

Program Address Subsection (P)

This subsection contains a 17-bit register which normally stores the address of the Next Instruction word (NI). The contents of this register are increased (P + 1) at a particular point in each instruction cycle. Thus the computer has the means by which it can initiate and govern the sequential execution of a program's instruction words.

When the instruction sequence is altered by a jump instruction, the address which replaces the current contents of P is the one to which program control is being transferred.

During the execution of instructions which operate in the Repeat Mode, this subsection stores the Repeat Count which is decreased (k-1) at each reiteration of the instruction.

Program Control Subsection (PC)

This subsection consists of two registers (F_0 and F_1) which decode instructions. In alternate-bank operation, the current Instruction Word as well as the NI word are decoded at the same time. The NI is loaded into F_0 while F_1 is decoding the current Instruction Word.

Storage Class Control Subsection (SCC)

The SCC subsection decodes the effective operand address, U, of an instruction for subsequent referencing to the Control Registers or main storage. Relative Addressing takes place within this subsection. (See Section 5 Relative Addressing.)

Arithmetic Section

Binary, Octal, and Decimal Notation - In discussing processor operations, frequent use of the octal and binary number systems is made. The processor uses only binary arithmetic. However, the programmers and operators who normally think in decimal terms must communicate with this binary machine. To avoid the use of cumbersome and error inducing strings of zeros and ones, they use octal notations as an intermediate shorthand. For example, the binary number

001	110	101	100	
3	6	5	4	can be expressed in octal as 3654_8.

In this manual, subscripts are used to identify the notation system in which the number is written. For example, 1964_{10} or 3654_8. When the meaning is clear from the context, the subscript is omitted.

During the execution of arithmetic instructions, temporary internal storage registers within the arithmetic section itself are used for the actual computations. The processor first determines that the arithmetic section will be used in a given operation. Data has been transferred automatically to a temporary internal storage register in the arithmetic section. The results of the arithmetic operation are then returned to one of the A-Registers. The internal storage registers cannot be addressed by the programmer. Thus, for all practical purposes, the A-Registers are treated as accumulators.

Adder - The adder in the UNIVAC 1108 Processor is a 36-bit ones - complement, subtractive adder (mod $2^{36} - 1$). Additions are performed in the following manner:

Assume that the value 2 is to be added to the value 6. The binary equivalents of these values are·

000 000 000 000 000 000 000 000 000 000 000 110 = 6

000 000 000 000 000 000 000 000 000 000 000 010 = 2

In execution, the adder first complements the value 2.

111 111 111 111 111 111 111 111 111 111 111 101 = 1's complement of 2.

Next, the adder subtracts the 1's complement of 2 from the value 6. The subtraction itself involves an "End-Around Borrow".

```
(1) 000 000 000 000 000 000 000 000 000 000 000 110 = 6
    111 111 111 111 111 111 111 111 111 111 111 101 = 1's complement of 2.
    000 000 000 000 000 000 000 000 000 000 001 001
```

End-Around Borrow

```
                                                   -1
    000 000 000 000 000 000 000 000 000 000 001 000 = 8
```

This method of adding is an internal process, and the programmer need not concern himself with it. For the purpose of analysis and debugging the programmer may manually simulate the computer operation by simple binary and octal addition.

BINARY		DECIMAL		OCTAL
110	=	6	=	6
010	=	2	=	2
1000_2	=	8_{10}	=	10_8

Addressable Registers (Control Registers) - There are 128 program-addressable, integrated-circuit, 36-bit registers which provide high-speed internal storage. The following paragraphs discuss each of the registers. Table 4-1 summarizes their addresses.

Address 000_8 (Processor State Word) - The contents of the Internal Function Register are stored at address 000_8 on the occurrence of an Interrupt.* The Internal Function Register is addressed only by the operation code 72-15 (Load Processor State Register). It cannot be addressed, for example, by the u-Field. The only way to examine the content of the Internal Function Register is via location 000_8 at the time of an interrupt. Location 000_8 is referred to as the Processor State Word.

Address 001 - 017_8 (Index X-Registers) - These 15 addresses are used as index registers. They normally are loaded with Index Words by means of a Load X_a (27, 0-17), a Load X_a Modifier (26, 0-17), or a Load X Increment (46, 0-17) instruction. The Index Words consist of an 18-bit modifier (X_M) in the lower half of the word and an 18-bit increment (X_I) in the upper half of the word. If the 18-bit upper half-word is loaded with a negative number, it works as a decrement.

35	X_I	18	17	X_M	00

X_I (18 bits) - Increment or Decrement
X_M (18 bits) - Modifier

The indexing operation performed during the execution of each instruction is accomplished as follows: three internal registers are provided to hold the u-Field, the modifier (X_M), and the increment (X_I). Each of these internal registers is 18 bits in length and is automatically cleared at the beginning of every machine cycle.

*Any interrupt, whether I/O, arithmetic, fault, or the execution of the instruction, Executive Return.

The contents of the register containing the modifier (X_M) and the u-Field are sent to the Index Adder in the Index Subsection. There the u-Field and (X_M) are added to form the effective operand address "U". (Relative Addressing is done here also. See Section 5.) This resultant sum is transferred to the Storage Class Control Subsection SCC. If the instruction calls for incrementation of the modifier (X_M) indicated by a "1" in bit position 17 of the instruction word, an additional step is included to add the contents of X_I to the contents of X_M in order to form a new modifier (X_M) value. This additional step does not lengthen the execution time.

NOTE· The u-Field of any instruction contains only 16 bits. Two additional "0-bits" are automatically filled in at the left before the Index Adder proceeds.

Address $014\text{-}033_8$ (Accumulator A-Registers) - Sixteen control registers located at addresses $014\text{-}033_8$ provide storage for arithmetic operands and results. The actual computation is performed in the arithmetic section with the results being stored in the Accumulators (A-Registers) specified by the particular instruction.

Because four of these locations overlap addresses assigned to Index Registers, the 1108 Processor is capable of additional versatility in index modification. For example, the result of a given calculation can for the next instruction be applied as a modifier to the u-Field.

The word format of the data to be loaded into an A-Register depends upon the type of arithmetic operation that is to be performed. For instance, if fixed-point arithmetic operands are used then they must conform to the format of the basic data word; floating-point arithmetic requires a word format of its own.

For those instructions using more than one arithmetic register, as do the floating point and scale factor instructions, addresses 034 and 035_8 may be used as the additional registers.

Address $040\text{-}077_8$ (Input/Output Access Control Registers) - The addresses $040\text{-}057_8$ are assigned as Input Access Control Words. They have the following format:

G	W	V
35 34	33 18	17 00

where G=the incrementation designator, W=the number of words to be transferred, and V=the initial main storage address to or from which data will be transferred.

The actual word-by-word transmission over an I/O channel is governed by the Access Control Registers. Two of these registers, one for input and one for output, are assigned to each of the sixteen channels. Input Access Control Registers govern the transmission of internally specified index (ISI) Input Data Words. The Output Access Control Registers govern transmission of both ISI Output Data Words and Function Words.

In initiating an input/output operation, the appropriate Access Control Word is sent to the Access Control Register which corresponds to the channel associated with the active peripheral unit(s).

The bit configurations in G and the operations specified by each are:

G	OPERATION	NEXT ADDRESS
00	Increment V Address	V + 1
01	Inhibit Increment	V
10	Decrement V Address	V - 1
11	Inhibit Decrement	V

The second use for the Input Access Control Word Addresses ($040-057_8$) is to store the externally specified index (ESI) Identifier Word. This is brought about because many communications peripheral devices are multiplexed (combined) into one I/O channel. This requires a method to provide for as many Access Control Words as there are pieces of communication equipment. To provide this information, each request originating from a communication peripheral device is accompanied by an ESI Identifier Word which addresses a reserved area of main storage containing many ESI Access Control Words. Each communication line can have its own reserved area of storage. Operation with the ESI Identifier Word given to the computer by the peripheral device is known as Externally Specified Index (ESI). After the computer has used the ESI Status Word to obtain the correct ESI Access Control Word, the word is stored in address $040-057_8$, (040 used for channel 0, 41 for channel 1, etc.) in the following format:

NOT USED	MSR	ESI Bias	CLT Number	NOT USED	MSR (2 bits)	ESI Bias (9 bits)	CLT Number (6 bits)
35	34 32	32 24	23 18	17	16 15	14 6	5 0

Input ESI Identifier Word — Output ESI Identifier Word

During input operations, the ESI Identifier Word (18 bits) is stored in the upper 18 bits of the Input Access Control Word location and during output operation the ESI Identifier Word is stored in the lower 18 bits of the Input Access Control Word location (040-057). (See Section 6.)

Address 100_8 (RØ - Real-Time Clock) -

	Clock Count
35 18	17 0

The Real-Time Clock is activated whenever the processor start button is depressed. Hardware circuits address the Real-Time Clock and decrease the count by one 5000 times per second without program control or supervision. A Real-Time Clock Interrupt occurs when the clock count has decreased through zero. If the Real-Time Clock is loaded with 5000, the Interrupt to main storage address 231_8 occurs in one second.

Address 101_8 (Repeat Counter) R1 -

Unassigned	k
35 18	17 00

k (18 bits) - Number of times instruction is to be executed.

The Repeat Counter is used to control repeated operations such as Block Transfer and search commands. To execute a repeated instruction "k" times, the repeat counter must be loaded with "k" prior to the execution of the instruction. A repeated sequence may be suspended to process an Interrupt with circuitry providing for the resumption of the repeated sequence after the Interrupt has been processed.

Address 102_8 (R2 Mask Register) - The Mask Register functions as a filter in determining which portions of words in masked operations or logical comparisons are tested. "U" is compared to "A" but only in those positions where "1's" exist in the Mask Register. In repeated masked search operations both the mask register and the Repeat Counter must be loaded prior to executing the actual search command.

Address 120_8 (Unassigned Exec. R-Register) - This location is not assigned a specific function, but may be used as a loop counter, a transit register, or to hold intermediate values as constants.

Address 121_8 (Exec. Repeat Count Register) - This register controls repeat operations such as Block Transfers and Repeat Search commands. This address has the same format and functions exactly the same as the normal Repeat counter (Address 101_8) except that when in the Exec. AXR Mode (D_6 equals one) this register is referenced instead of Address 101_8.

Address 122_8 (Exec. Mask Register) - This register functions as a filter to determine which portions of words in mask operations are to be used. This address functions the same as the normal Mask Register (Address 102_8) does for user programs except when the D6-Designator (Exec. AXR Mode) equals one. Then, this address is referenced rather than Address 102_8.

Address $123\text{-}137_8$ (Unassigned Exec. R Registers) - These registers are not assigned specific functions but may be used by the Executive as loop counters, transit registers, or to hold intermediate values or constants.

Address $140\text{-}157_8$ (Exec. Index Registers) - These sixteen addresses are to be used as Index Registers in the same fashion as the Index Registers at Addresses $001\text{-}017_8$. These Registers are used when the Exec. AXR Mode is equal to one and X-Register references are made.

Address $154\text{-}173_8$ (Exec. Accumulator A-Register) - These sixteen Registers are used to store the arithmetic operands and results. These are used instead of Addresses $014\text{-}033_8$ when the Exec. AXR Mode is equal to one.

Address $174\text{-}177_8$ (Exec. Unassigned Registers) - These four registers are not assigned specific functions but may be used as required.

The 128 words of control registers provide multiple index-registers, multiple accumulators, input and output Access-Control locations for each of the sixteen I/O channels, an addressable Real-Time Clock, and a complete set of Index Registers, Accumulators, and R-Register for the Executive Routine.

The control registers allow the control circuits in the arithmetic section, index-adder, and input/output section to reference the contents of the control registers several times during the execution of a single instruction.

Table 4-1. Control Register Address Assignments

OCTAL ADDRESS	REGISTER SIZE IN BITS	FUNCTION
000	1	Processor state Word
001 to 017	15	Index (X) Registers
*014 to 033	16	Accumulator (A) Registers
034 to 037	4	Unassigned
040 to 057	16	Input Access Control Words
060 to 077	16	Output Access Control Words
100	1	RØ Real-Time Clock
101	1	R1 Repeat Count Register
102	1	R2 Mask Register
103 to 117	13	Unassigned R-Registers
120	1	Unassigned Exec. R-Register
121	1	Exec. Repeat Counter Register
122	1	Exec. Mask Register
123 to 137	13	Unassigned Exec. R-Registers
140 to 157	15	Exec. Index (X) Registers
*154 to 173	16	Exec. Accumulator (A) Registers
174 to 177	4	Unassigned Exec.

*Addresses 014 to 017 and 154 to 157 are dual-purpose ("overlapping") registers.

Main Storage

The 1108 System is designed with a modular main storage permitting a variety of storage sizes to be used. Starting with a storage of 65,536 (36-bit) words distributed in two 32,768-storage modules expansion is possible (in increments of 32,768 words) to a maximum of 262,144 words in eight independent 32,768-word storage modules.

	Module 1	Module 2	Module 3	Module 4
Option 1	0-32,767	32,768-65,535		
Option 2	0-32,767	32,768-65,535	65,536-98,303	98,304-131,071

The primary advantage of using two storage modules is that it is possible to store data in one module and instructions in another. References to consecutive instructions can then be overlapped to save time; that is, the current instruction need not be fully executed before the next instruction is being called up. This results in an effective cycle time of 375 nanoseconds whereas if both the data and instruction are stored in the same module, the cycle time is 750 nanoseconds.

Any attempt to retrieve data from an address beyond the capacity of main storage results in a word of all 0's unless the computer is in the Guard Mode. In the Guard Mode an Interrupt is generated to Address 243_8 in main storage. If an attempt is made to store data in an address beyond the capacity of main storage and the processor is not in the Guard Mode, the instruction is executed in all respects except that no actual writing of data takes place.

The first 128 (177_8) addresses have the same address as the control registers. They can be referenced only by indirect addressing (using the i-field) or as instructions when control is transferred to this area by a jump instruction.

An input/output instruction, that is, one which specifies transfer to or from a peripheral equipment will reference only main storage. Therefore, when the contents of the control registers ($000-177_8$) are to serve as output data, they must first be transferred by the program to main storage. This safety feature refuses input/output data transfers direct access to the control registers, and thus, eliminates the possibility of inadvertent destruction of the data stored therein.

Because the lower addresses of main storage ($000-177_8$) are somewhat protected, they are particularly well suited for containing part of the Bootstrap Routine (Addresses $000-200_8$).

Main Storage Protection - The instruction 72-16 (Load Storage Limits Register) is used to establish the two usable areas of storage that may be referenced by a program. All other locations in storage not so designated can be locked out by operating in one of the two following modes·

1) Guard Mode (Read, Write and Jump Protection)
2) Write Mode (Write Protection)

The selection of one of these two modes of protection is made by the master bit selection in the Processor State Register described later in this section.

Main storage is divisible into increments of 512_{10} words. This means that the permissible area may be 512_{10}, $1,024_{10}$, or 1536_{10} words in length (etc.).

The $262,144_{10}$ words of storage can be thought of as being divisible into 512_{10} increments, each consisting of 512 words. Thus, 512 decimal numbers equals 777 octal numbers or 111 111 111 in binary notation. Therefore, if the first 512-word increment is assigned the binary notation of 000 000 000 and second 512-word increment assigned the 000 000 001 notation, we can specify any 512-word increment of the 262, 144 words of storage (512 increments) by using only 9 bits.

Main storage also has addresses $000\text{-}177_8$, but these are accessible only through indirect addressing, indexing, or as instructions.

The two usable storage areas are determined by <u>two sets</u> of limits (one upper limit and one lower limit for each area). The two areas may be separate, partially overlapping, or completely overlapping.

To load the 36-bit Storage Limits Register, the instruction 72-16 (Load Storage Limits Register) is used. The operation of this instruction is as follows:

The contents of U ⟶ Storage Limits Register:

For example, any location specified by the u-Field of this instruction may be referenced for purposes of loading the Storage Limits Register.

The format of the 36-bit Storage Limits Register after loading is shown below:

Upper Limit (8 bits)	Lower Limit (8 bits)	Upper Limit (8 bits)	Lower Limit (8 bits)
34 27	25 18	16 9	7 0
I Area		D Area	

Execution of the Load Storage Limits Register instruction (72-16) does does not enable the storage protection feature; it merely loads the storage limits/values. To institute storage protection, the command 72-15 (Load Processor State Register) must be executed. With the loading of the Processor State Register, a selection of one of the two modes of protection (Guard Mode (D_2) or Write Protection Mode (D_3)) will be made by bit selection of the designators in the Processor State Word as shown below:

D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	BI	2	BS	BD
Floating-Point Zero	Base Register Suppression	Control Register Selection	Double-Precision Underflow	1107 Compatibility	Write Only Storage Protection	Guard Mode	Over-Flow	Carry	8 bits	not used	7 bits	8 bits
35	34	33	32	31	30	29	28	27	25 18	17 16	15 9	7 0

FLOATING POINT ZERO

BI, BS, and BD are described in the section on Relative Addressing.

D_0 = Carry Designator

D_1 = Overflow Designator

D_2 = Guard Mode Designator. When this bit is a 1, Guard Mode will be activated and the processor will not read, write or jump outside the two areas set by the Storage Limits Register. Also, this will not allow the 1108 to execute certain instructions without causing an Interrupt to core memory location 243_8 (see the Guard Mode Section). When the D_2 Designator is a 0, the Guard Mode is not enabled.

D_3 = Write-only Storage Protection Designator. When this bit is a 1, the Write Mode will be enabled. When the Write Mode is enabled, the 1108 cannot write outside the two areas established by the Storage Limits Register except CR address $0\text{-}37_8$ and $101\text{-}117_8$ without causing an interrupt to main storage location 243_8. When this bit is a 0, the write mode is not enabled.

The D_3-Designator and D_2-Designator are interactive as follows:

D_2, D_3 = 00 no Guard Mode and no Write Protection (read, write, and jump anywhere).

D_2, D_3 = 01 Guard Mode enabled (read, write, and jump; memory protection and Interrupt on the occurrence of certain instructions. See Guard Mode Section for list of instructions).

D_2, D_3 = 10 Write-only Storage Protection, no Guard Mode (read and jump anywhere).

D_2, D_3 = 11 Guard Mode enabled with Write-only Protection. (Write Protection but read and jump anywhere; interrupt on the occurrence of certain instructions).

D_4 = 1107 Compatibility Designator. When this bit is 1, the upper 2 bits are cleared from the absolute address, S. This eliminates addressing in the range of $200,000_8$ - $377,777_8$. In this mode, only the first 65K of storage can be used.

D_5 = Double-Precision Underflow Designator. This bit controls the Interrupt action when Characteristic Underflow occurs in double-precision floating-point arithmetic. When $D_5 = 0$ and a Characteristic Underflow occurs, the instruction at main storage location 245_8 is executed next and original contents are not disturbed. When $D_5 = 1$, Characteristic Underflow will not cause an Interrupt but instead the result is written back as 72 zeros and the normal instruction sequence continues.

D_6 = Control Register Selection Designator. When $D_6 = 1$, the Exec AXR Register (CR addresses $120\text{-}177_8$) are used instead of the normal A-, X- and R-Registers. Now the x-Field and the a-Field which normally reference CR addresses $001\text{-}017_8$ and $014\text{-}033_8$ respectively, reference CR addresses $140\text{-}157_8$ and $154\text{-}177_8$ respectively. Also, now the 23, 0-17 (Load R) instruction which normally used an a-Field to reference the R-Registers and CR addresses $100\text{-}117_8$ now references CR addresses $120\text{-}137_8$. Any reference to the Exec AXR-Registers using the u-Field or j- and a-Fields must reflect the exact CR address. When $D_6 = 0$ the normal AXR-Registers are referenced.

D_7 = Base Register Suppression Designator. When $D_7 = 1$, the indirect bit (i-Designator) of an instruction will not be interpreted for indirect addressing. Rather, it will be used to cause the storage address "S" to be developed from the u-Field + X_M and NOT from the u-Field + X_M + BI, or from the u-Field + X_M + BD as in the normal case. (For definition of BI and BD, see section on Relative Addressing.)

D_8 = Single-Precision Floating Point Designator. If D_8 equals zero and the resulting mantissa is zero, then the most significant word is set to all zeros; the least significant word is not affected. If D_8 does not equal zero and the resulting mantissa is zero, the most significant characteristic will not be set to zero.

Upon the appearance of any Interrupt, storage protection becomes inactive. The contents of the PSR are stored in CR Address 000. The Processor State Register (PSR) designators are all cleared to zeros with the exception of D_6 and D_7 which are set to one, and control is passed back to the Executive routine. Note, however, that neither loading the PSR nor the occurrence of an Interrupt has any effect on the values of the Storage Limits Register. These values are influenced only by the instruction 72-15 (Load Storage Limits Register).

Input/output is not affected by the storage protect feature. That is, data may be written into or read from any main storage location regardless of the setting in the Storage Limits Register or the Guard Mode and the Write Mode.

The address coming from the P-Register is not checked against the values that are stored in the Storage Limits Register. Since the address coming from the P-Register is not checked against the values that are stored in the Storage Limits Register, the 1108 Processor has no hardware that can determine when the end of a user program is reached. If control is not turned over to the Executive Routine at the end of a user program, the P value which is equal to the address of the last instruction (or data) is increased by one and the next address (P + 1) is read into the Program Control Subsection. This address will contain an instruction (or data) of the next program, but in either case the processor will execute it. One way (there are others) to eliminate

this problem during assembly with the 1108 assembler is to include a 72-11 (Executive Return) instruction as the last instruction of a user program.

Guard Mode

In addition to the write, read and jump protection described in the main storage protection section of this manual, certain instructions cannot be executed when the processor is in the Guard Mode. These are as follows:

72-15 Load Processor State Register
72-16 Load Storage Limits
73-14 Initiate Inter-Processor Interrupt
73-15 Select Interrupt Locations
73-16 Load Channel Select Register
75-(0-17) All I/O Instructions
A Write in any location 40_8 through 100_8 and $120\text{-}177_8$
Lockout Interrupts for more than 100 microseconds

Note that when the computer is in the Guard Mode it still is possible to use the 72-13 (Prevent all I/O Interrupts And Jump) instruction. This is necessary so that the special, customer-written, real-time I/O programs can effectively run under the Guard Mode. However, when in this mode, Interrupts can only be locked out for more than 100 microseconds.

Activating the Guard Mode is explicit in the instruction 72-15 (Load Processor State Register). Execution of this instruction is the only way that the Guard Mode can be made operative. When in Guard Mode, an attempt to perform any of the invalid functions listed above causes an Interrupt to core memory location 242_8.

Sequence of Operations (Functional)

To illustrate the functions of the various Instruction Word designators, assume that an arithmetic instruction stored at the address contained in the P Subsection is to be executed. Assume further that instructions are stored in one storage module and data in the other. The j-Designator value is not equal to 16 or 17, and the x-Designator is not equal to 0. Figure 4-2 illustrates the major subsections involved in the following sequence.

Once the arithmetic instruction has been read into PC Subsection, the following events take place·

1) The f-, j-, and a-Designators are interpreted and the appropriate circuitry is alerted.
2) The lower half of the instruction (h-, i-, and u-designators) is transferred from PC Subsection to the index unit.
3) The x-Designator is tested to determine which index register, if any, is to control address modification.

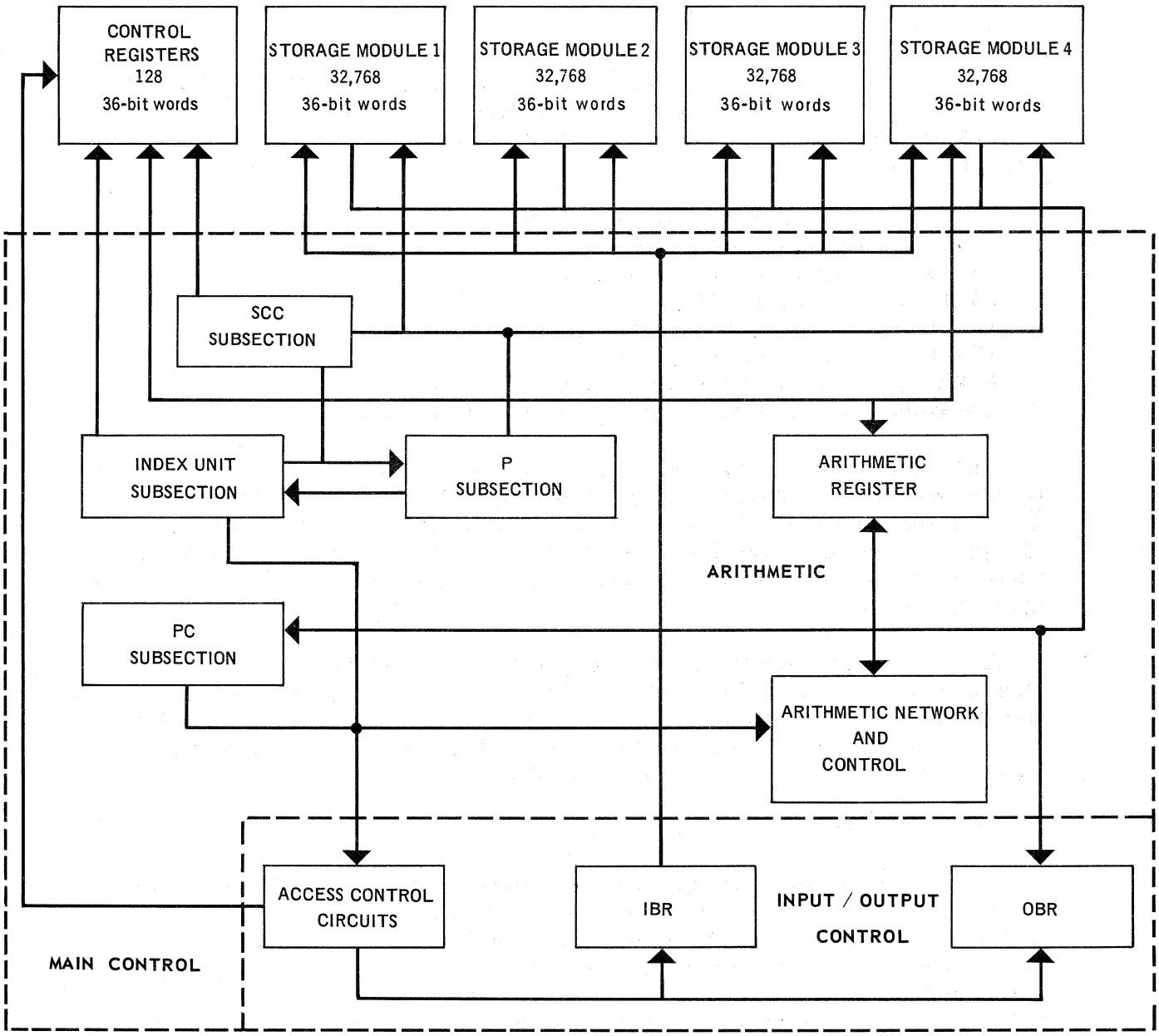


Figure 4–2. Control Section Block Diagram.

4) If modification is stipulated (the contents of "x" are unequal to 0), the lower half of the contents of the specified index register is transferred to the adder in the index unit.
5) The contents of the u-Designator, with two 0's placed to the immediate left, are transferred to the adder where modification takes place as 18-bit one's complement addition.
6) After index modification takes place, the address is tested to see if any of the following conditions exist:

   u 200 and i = 0
   f 70, b = 0, j = 16 or 17
   f 70, b ≠ 0, j = 16 or 17
   u-Field = shift count

6a) BI or BD (for definition, see section on Relative Addressing, section 5) is added to form the storage address S.
7) After the addressing takes place, the address is transferred from the adder to the SCC Subsection where it is decoded for subsequent referencing of storage.
8) The i-Designator is tested to determine whether direct or indirect addressing is stipulated.
9) When modification is specified, the h-Designator in the current instruction is tested to determine whether the index register modifier (X_M) is to be increased (or decreased) by (X_I). If "h" equals 1, the increment is applied to the modifier.
10) After incrementation, the new modifier is sent into the lower half of the index register specified by the x-Designator. The increment portion remains unchanged.
11) The operand address is transferred from SCC Subsection (Step 7) to the appropriate address selectors.
12) The entire 36-bit contents of the location specified in the storage address register are transferred into the Z-Register of the appropriate storage unit.
13) The contents of the A-Register specified in the current instruction are transferred from CR's to an arithmetic X-Register.
14) The actual data transfer, in accordance with the j-Designator interpreted in Step 1, is made from main storage (Z1, Z2, Z3, or Z4) to the arithmetic section. *
15) The Program Address Subsection, P, is increased by 1 to provide for the Sequential execution of instructions.
16) The next instruction stored at the address now contained in P is referenced in storage.
17) The circuitry alerted by the f-Designator in step 1 performs the desired arithmetic operation.
18) The next instruction, referenced in step 15, is sent to PC Subsection.
19) An input/output transmission may be performed while the specified arithmetic operation (step 17) is being completed.

*The j-Designator is ineffective when the operand is read from CR's (Z_0).

For most instructions, the preceding steps require .75 microseconds. Execution time is extended by .75 microseconds when the operand reference is made to the same bank as the instruction reference.

j-Designator Unequal to 16 or 17 - When the j-Designator of the current instruction is neither 16_8 or 17_8, one of eleven possible portions of a word or the entire word may serve as the operand. The j-Designator becomes effective when the operand is being transferred between main storage (Z1, Z2, Z3, or Z4) and the arithmetic section (Figure 4-3). Consequently, with respect to the j-Designator, a data transfer to or from CR's always involves a complete 36-bit word.

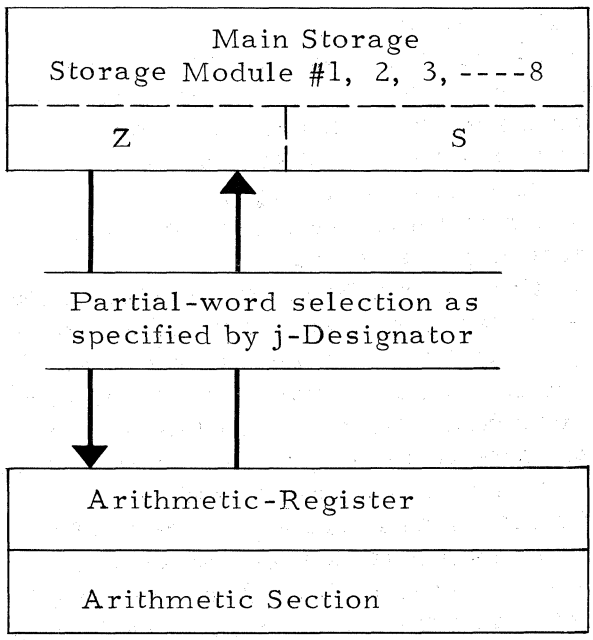


Figure 4-3. Data Paths for j not equal to 16 or 17

j-Designator Equal to 16 or 17 - The j-Designator may also stipulate that the operand is to be transferred from the Instruction Word itself. This operation is specified by a j-value of 16 or 17_8. Then, depending upon the contents of the x-Designator, either 16 or 18 bits will be transferred from low-order positions in the current instruction to the arithmetic section.**

When j equals 16 and x is not 0 (that is, index register modification is specified), the 16-bit contents of the u-Designator of the current instruction serve as the operand. In the index unit, two binary 0's are placed to the immediate left of the 16 bits taken from the instruction. After the specified modification has been performed as 18-bit, one's complement addition, the 18-bit operand is transferred to the lower half of the arithmetic Register for subsequent transmission to the arithmetic section. The upper half of this register is cleared to 0's (see Figure 4-4).

**The j-values of 16 and 17 are effective only in transfers <u>to</u> the CR's. These values inhibit transfers <u>from</u> the CR's.

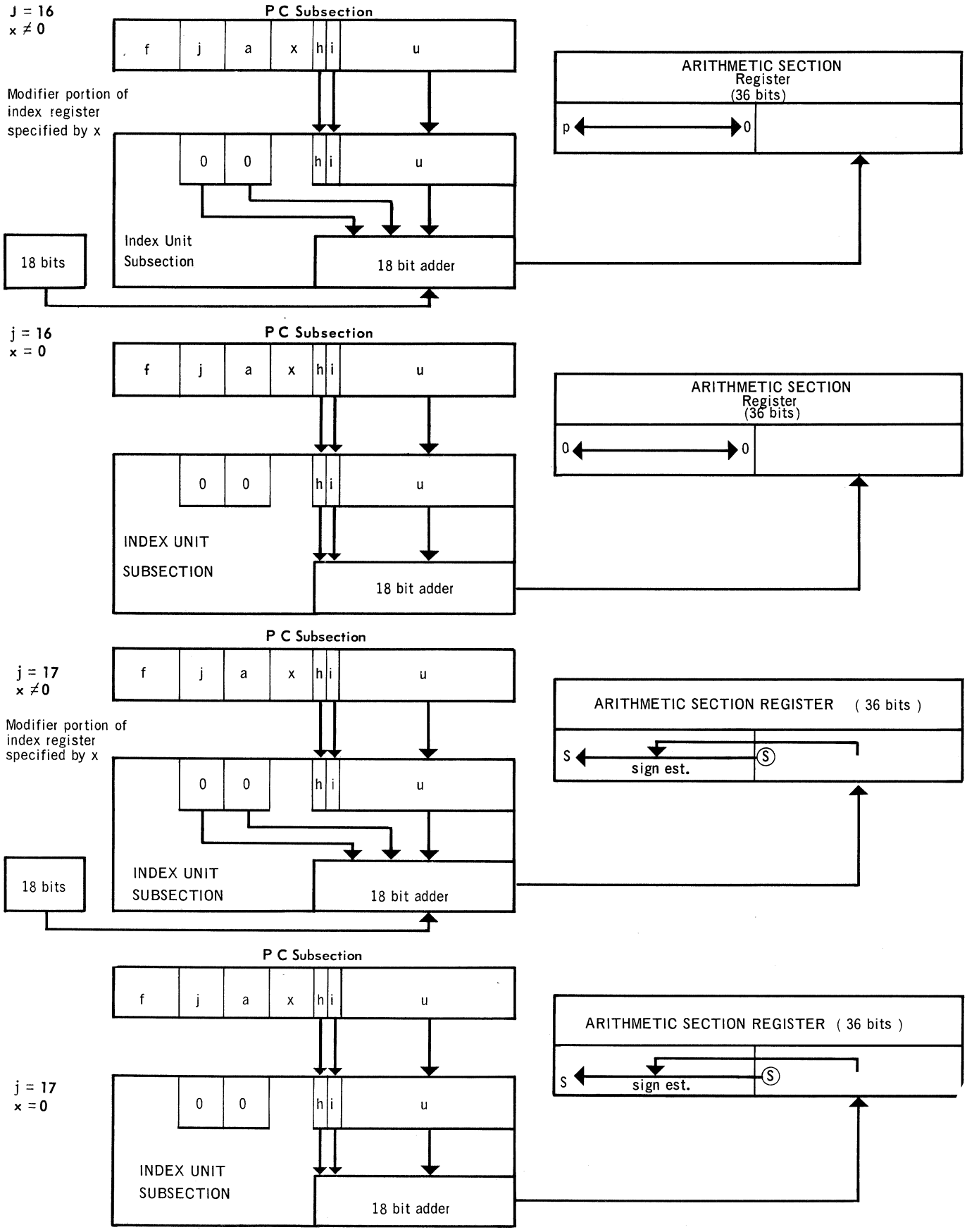
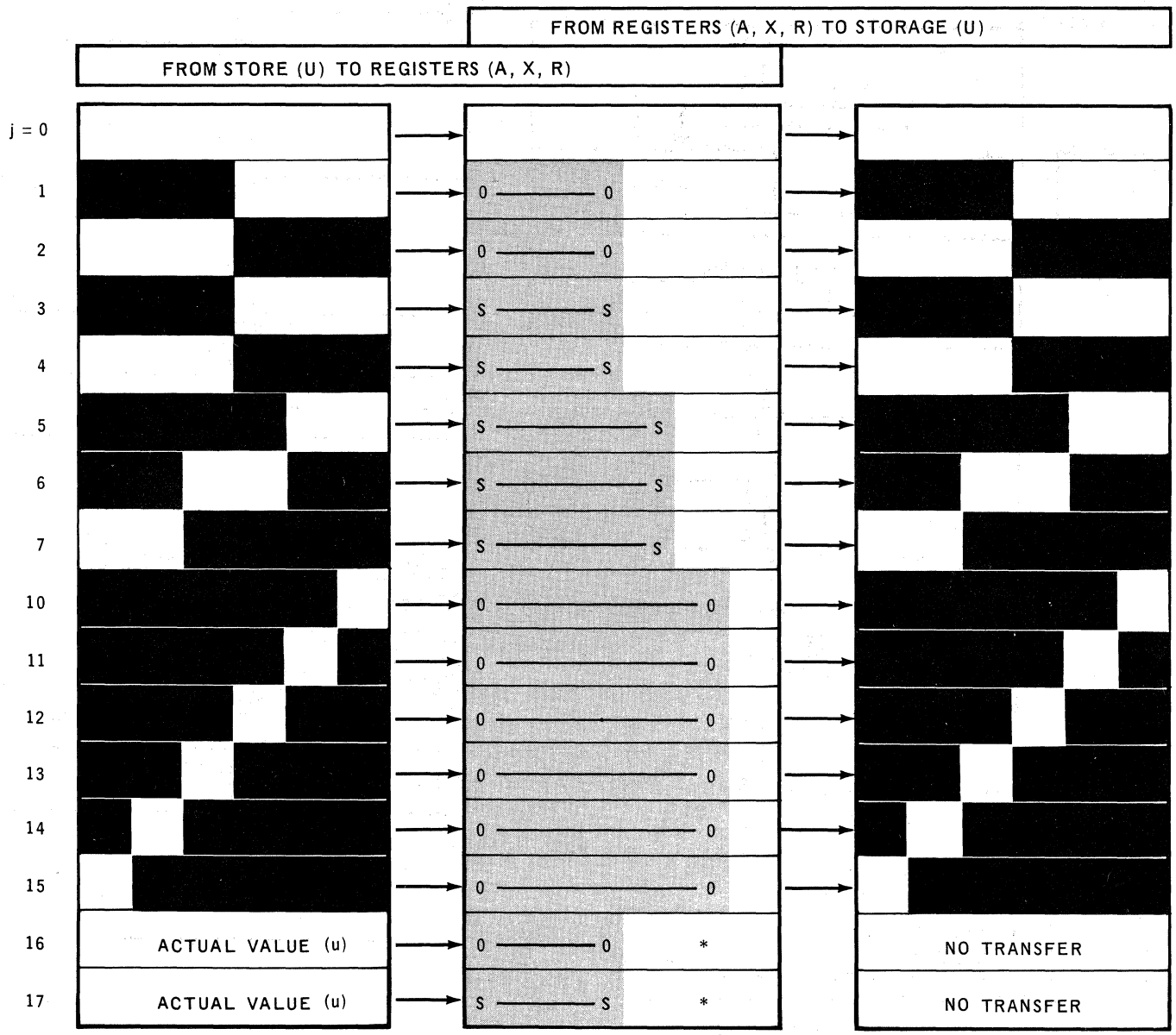


Figure 4–4. Data Paths For A "j" Equal to 16 or 17.



= Portion of word involved in the operation

= Portion of word not involved in the operation

= Remainder of word in arithmetic register
0 — 0 = zero fill (may be plus or minus zero)
S — S = sign fill (sign of j-determined final contents of A)

* If x-designator = 0, the transferred value = h, i, u
If x-designator ≠ 0, the transferred value = u + $(X)_0$

Figure 4–5. J–Determined Partial Word Operations.

When j equals 16 and x equals 0, modification is inhibited and an 18-bit operand is transferred from the instruction word to the arithmetic section. The 18 bits are taken from the h-, i-, and u-Designators of the current instruction.

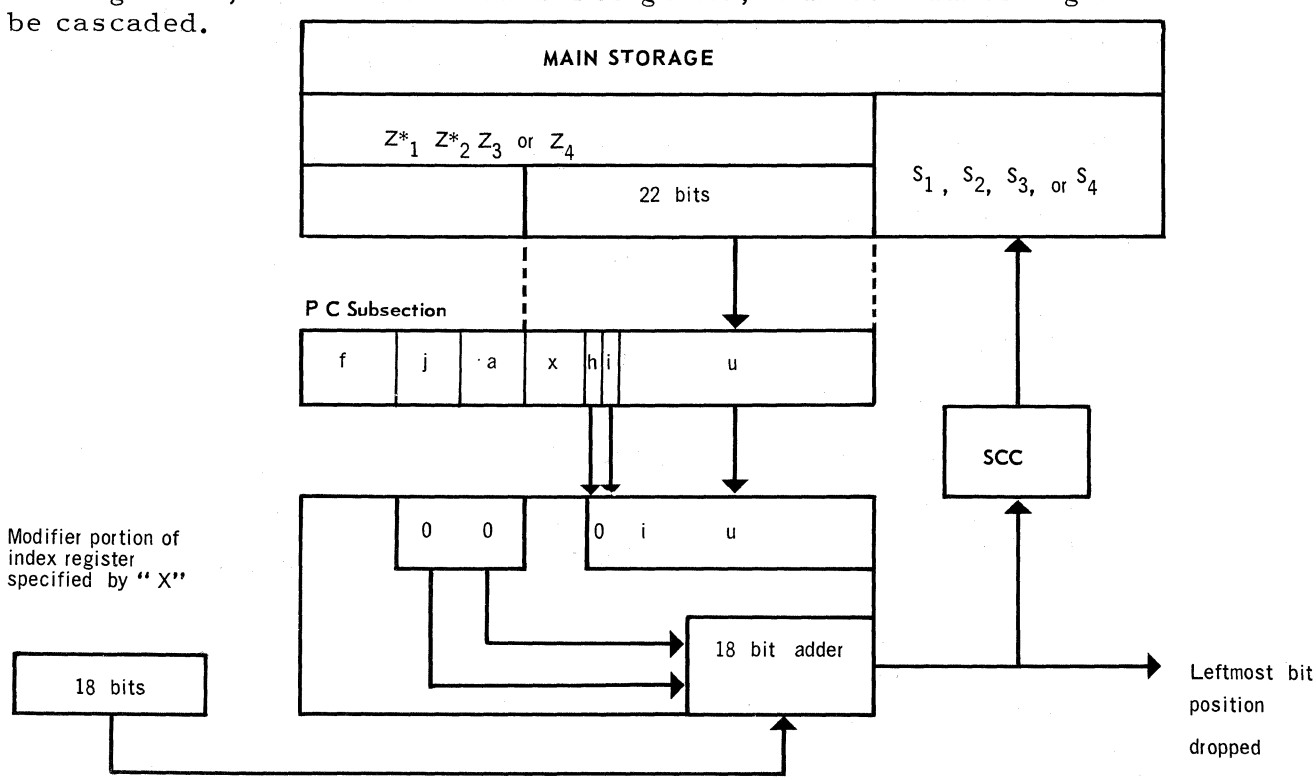
When j is 17 instructions are executed in a manner similar to those described above for j equals 16 with the exception that the sign of the operand as it enters the lower half of the X-Register is extended to the left. Sign extension, then, replaces the filling in of 0's.

Once the operand has entered the arithmetic section, the specified operation is performed. Indirect addressing, if specified, is performed before the transfer. Figure 4-5 summarizes the effect of the j field on partial-word transfers.

Indirect Addressing - When the i-Designator of the current instruction is equal to one, an indirect addressing operation will be performed. In this case, the rightmost 22 bits contained in Z_1, Z_2, Z_3, and Z_4 are transferred (step 11 in the execution cycle) to corresponding positions in PC Subsection. The execution cycle then reverts to step 2 and remains in this loop until step 12 specifies direct addressing.

Note that the 22 bits cannot be transferred from control register (Z_0) since the PC Subsystem can only be entered from main storage, Z_1, Z_2, Z_3, or Z_4. (See Figure 4-6.)

Because the 22 bits read into this subsection include the x-, h-, and i-Designators, as well as a new u-Designator, indirect addressing can be cascaded.



* Z may contain an Instruction Word or a Data Word

Figure 4–6. Data Paths for Indirect Addressing.

Definition of Overflow

Overflow is the generation of a bit of significance in the sign bit position. Reducing all the instructions to an addition (or subtraction which is complementing and adding), the overflow condition can be predicted by the following rules:

Adder inputs	Result Pos.	Result Neg.
(A) pos. & (U) neg.	clear	clear
(A) neg. & (U) pos.	clear	clear
(A) neg. & (U) neg.	set	clear
(A) pos. & (U) pos.	clear	set

The overflow indicator (bit D_0 in Processor State Register) is set when the quantities have like signs but produce a result of opposite sign.

Definition of Carry

Reducing all situations to the addition case, the carry indicator is set when an end-around carry is generated. The condition of the carry indicator (bit D_1 of PSR) after an addition can be predicted by the following rules:

Adder inputs	Result Pos.	Result Neg.
(A) pos. & (U) neg.	set	clear
(A) neg. & (U) pos.	set	clear
(A) neg. & (U) neg.	set	set
(A) pos. & (U) pos.	clear	clear

Special Cases For Carry

The following cases are worthy of noting as they all set the carry designator.

1. Any number added to its complement.
2. Any number added to all ones.

Using the carry or overflow jumps will not affect the states of the indicators. Thus, for example, any number of overflow jumps could be programmed to sample the same state of the indicator.

The instructions that condition both the carry and the overflow designators are·

AA	(14) Add to A
AMA	(16) Add Magnitude to A
AU	(20) Add Upper
AX	(24) Add to X
DA	(71, j=10) Double-Precision, fixed-point add.
ANA	(15) Add Negative to A
ANMA	(17) Add Negative Magnitude to A
ANU	(21) Add Negative Upper
ANX	(25) Add Negative to X
DAN	(71, j=11) Double-Precision, fixed-point add, negative.

Initiation of any of the above 10 instructions clears both the indicators. After the addition, if an overflow has occurred, the overflow indicator is set, and if an end-around carry has occurred the carry indicator is set. These two indicators remain in this state (set or cleared) until another one of the 10 affecting instructions is initiated.

5. RELATIVE ADDRESSING

Program Segmentation

1. Programs must be written in two portions to run in the most efficient manner on the 1108 System. Each portion may be all instructions, all data or a mixture of data and instructions. But the intermix of instructions and data must be written so that an instruction in one portion references data in the other portion. This does not have to be strictly adhered to for the program to run because programs (Instruction and Data) may be written as one continuous program.

2. When a program is written in two portions, one portion is the I-Area and the other portion is the D-Area. The I-Area may be the Instruction portion, the Data portion, or the intermix portion. The name, I-Area, is relative and only denotes what to call it; it is not necessarily the Instruction portion. Likewise, the D-Area is also a relative name.

3. The I-area of a program should be placed in one module of storage and the D-Area of a program should be placed in a completely different module of storage. This must be done to use overlapping storage modules.

General Theory of Relative Addressing - In the normal addressing sequence for the 1108 Processor, the u-Field is modified by one of the Index Registers, thus forming the effective address, U. Simultaneously the effective address is modified by a Relative Address Modifier BI or BD to form the storage address S. The last modification is called Relative Addressing.

During the last operation, the effective address U and a Relative Address Modifier (BI or BD) are added to compute the address of the desired information (Data or Instruction).

There are only two Relative Address Modifiers, BI and BD. BI is a Relative Address Modifier for the I-Portion of the program, and BD is the Relative Address Modifier for the D-Portion of the program. Because these two Relative Address Modifiers (BI and BD) remain constant for an entire program, the 1108 can relocate an entire program anywhere in main storage. For example, a program (and/or its data) may be

moved anywhere in main storage and then run without modification except for changing the Relative Address Modifiers (BI and BD). The moving of a program is not under the control of the individual worker program but under the control of the Executive Routine. Actually, users may not be aware even that the Relative Addressing operation is taking place. Therefore, the reading of this section of the manual by programmers is not necessary except for information purposes.

The Relative Address Modifiers BI and BD are stored in the Processor State Register (PSR), and changing of the Processor State Register by the worker program is prohibited by the operation of the Guard Mode.

To illustrate: Assume that main storage is loaded only with the Executive Routine and the necessary library routines. After the worker program has been loaded, the Executive Routine executes a 72-15 (Load PSR) Instruction to activate the Guard Mode. Once the Guard Mode is active, the 72-15 Instruction is disabled; therefore, if a worker program tries to execute the 72-15 Instruction, the 1108 interrupts the program and executes the instruction at Address 243_8. When any Interrupt occurs, the contents of the Processor State Register are stored (as the processor state word) at Address 000_8 of the control registers. At this time all positions in PSR are cleared except for D_6 and D_7; thus, the Guard Mode is not in effect and the subroutine that is associated with the Interrupt can execute a 72-15 Instruction and set the Processor State Register as needed for the subroutine.

<u>The Relative Addressing Modifiers</u> - BI and BD must be computed every time a program is loaded or relocated in main storage. To understand how these two Relative Address Modifiers are computed, we must first understand the organization of main storage. Each storage module comprises 32,768 (36-bit) words or 64_{10} (100_8) blocks. Each block consists of 512_{10} ($1,000_8$) 36-bit computer words.

If any words of the D or I area are placed in a block, the unused part of that block cannot be used to store another program if guard mode is desired.

Relative Addressing Format - The format of BI, BD, and BS appears in the Processor State Register as the following:

D_8	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	BI	Not to be used	BS	BD
									8 bits		7 bits	8 bits
35	34	33	32	31	30	29	28	27	25 18	17 16	15 9	7 0
Designator Section										Relative Addressing Section		

The Designator section is explained in the Guard Mode Section.

The Processor State Register is stored as the processor state word at Address 000_8 in the control registers upon the occurrence of an Interrupt. After the contents of the Processor State Register (PSR) are stored at 000_8, D_0, D_1, D_2, D_3, D_4, D_5, and D_8 are cleared and the D_6 and D_7 Designators are set to one.

When the interrupting program is finished, the contents of CR address 000_8 can be reloaded into the Processor State Register (PSR) and control transferred to the interrupted program.

Relative Addressing Sequence of Operation - To understand how the BI, BD, and BS are used in relative addressing, one must first review the section "Sequence of Operations" (Functional)", section 4. Note that in it step 5, the effective address U is completely formed. After that the following conditions are checked in step 6.

1. $U < 200$ and $i = 0$
2. $f < 70$, $b = 0$, $j = 16$ or 17
3. $f < 70$, $b \neq 0$, $j = 16$ or 17
4. Effective address U = shift count

If any one of these conditions exist, the operation continues to step 7; but if none of the above conditions exist (as will be the usual case), the effective address, U, is compared to BS to determine which is the larger. The comparison takes place in the following manner;

17 16	15 9	8 0

Effective Address U

Not to be used 2 bits 17 16	7 bits 15 9	9 bits 8 0

BS

Remember that each bit of BS represents one storage block or 512_{10} ($1,000_8$) words. Therefore, when BS = 001, it must be compared to the effective address, U bit position 15 through 9, to find the proper magnitude. The lower positions, 8 - 0, are not to be compared with BS. If U_{16} is a "1", then U is automatically larger than BS. If U_{16} is a 0, then the comparison of U_{15-9} to BS is made to determine which is the larger.

Two sums are formed for every instruction. (See Figure 5-1.) If the conditions checked in step 6 exist, the sums are discarded. If the conditions do not exist, one of the resultant sums (depending on whether BS is larger or smaller than U) is used to address main storage instead of the effective address, U. Again remember that 001 in BI or BD represents one storage block or 512_{10} ($1,000_8$) words. Therefore, any addition of BI or BD must be made with bit positions 15 through 9 of the U-Fields.

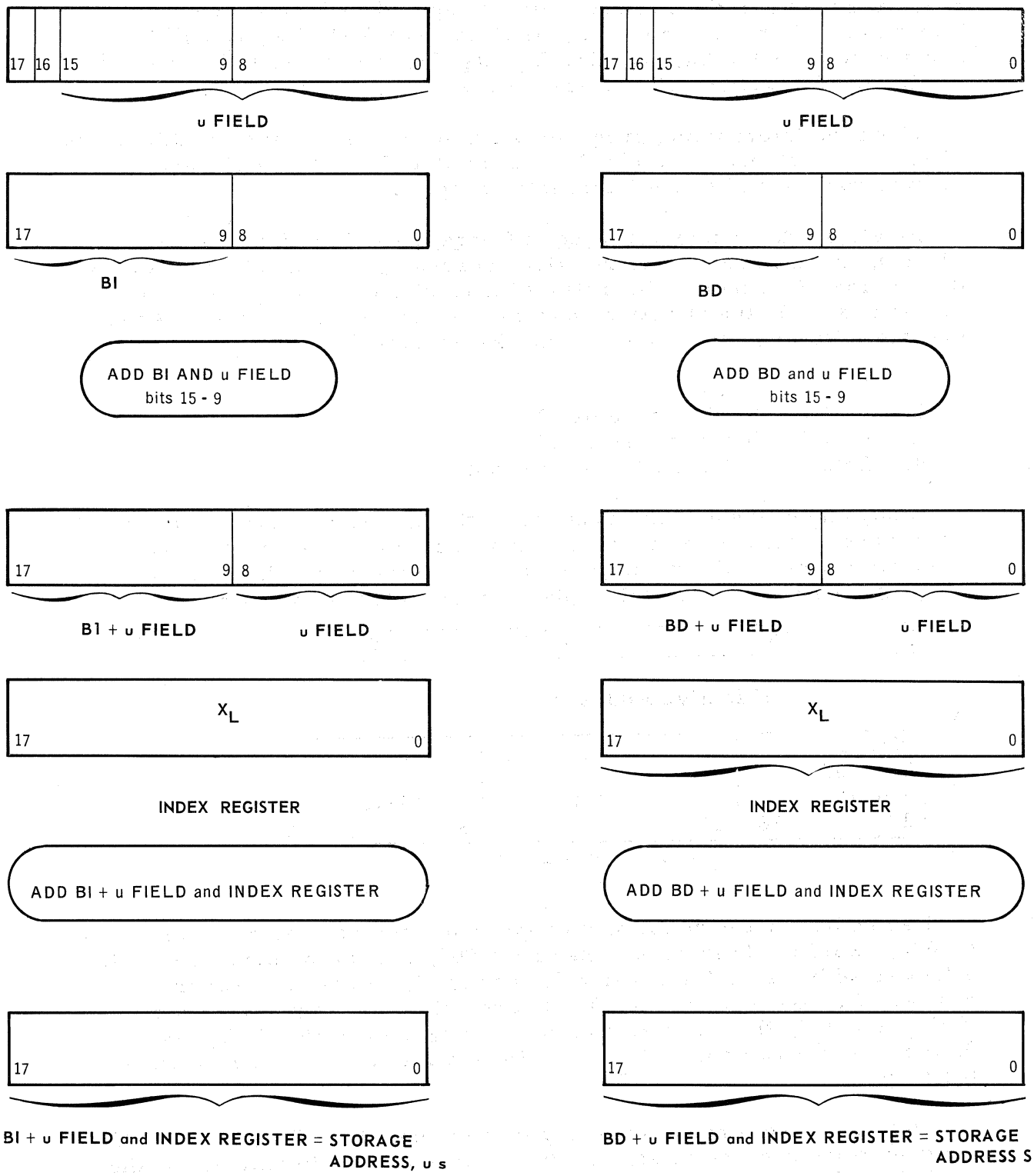


Figure 5–1. Formation of Storage Address.

After the first addition is completed, the Index Register as specified by the x-Designator is added to complete the two resultant sums (Absolute Addresses).

Now if BS is less than the effective address U_{15-9}, the resultant sum of BD + u-Field + Index Register will become the operand address and will be sent to main storage. The other resultant sum (BI + u-Field + Index Register) will not be used. If BS is greater than the effective address U_{15-9}, the resultant sum of the BI + u-Field + Index Register will become the operand address sent to main storage. The other resultant sum (BD + u-Field + Index Register) will not be used. Regardless of which resultant sum is used, it is always checked against the storage limits (I-area or D-area). The entire process of forming the sum and checking the storage limits is summarized in Figure 5-2.

General Summary; The Relocation Base Registers

The primary function of the base registers is to provide a relocation capability to any absolute (binary) program resident in main storage; i.e., the program (and/or its data) may be moved about anywhere in storage and then run without modification to the program. Controlling these registers and exercising relocation is not within the province or cognizance of the individual programs. Indeed, programmers need not be aware of such registers and facilities during the construction of their programs; the loading and manipulation of the base registers (or relocation registers, as they are sometimes called) is the exclusive domain of the Executive routine. Normal programs are prohibited from tampering with these registers by virtue of the operation of the Guard Mode. The procedures for computing the values for the base registers will be shown in detail later. However, in general, each computed main storage address generated by the program is subjected to the addition of one of the two base registers. Which one of the registers is added for each individual address is decided on the basis of the value in the switch register. The value in this switch register is computed in the Executive routine and reflects the length of the program. The values in the two extra index registers are a function of the length of the program, and of the absolute addresses in which program and data will be stored.

The Processor State Register

The 36-bit processor state register (PSR) holds the contents of the three registers used in base indexing as well as several special designators. This register is loaded by the instruction, Load PSR (f = 72, 15). For loading into PSR, the contents of any location in either main storage or the control registers may be specified by the u-Field of this instruction. Upon interrupt, the contents of the PSR are automatically stored into (000) processor state word.

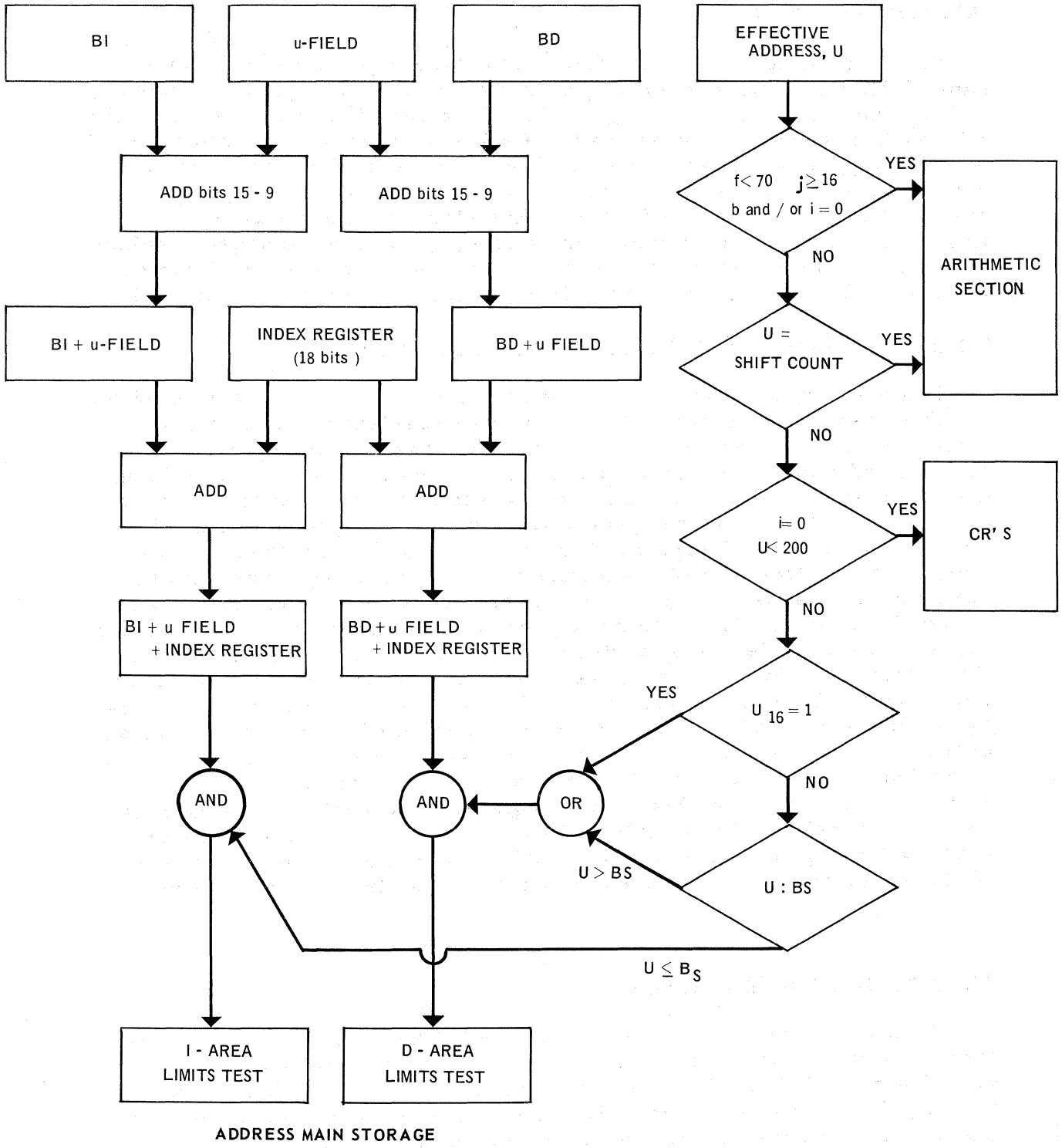


Figure 5-2. Relative Addressing Sequence

PSR has the following format ·

9	8	2	7	8
D	BI	NOT USED	BS	BD

BI, & BD are 8 bits each; their function will be described later.

D is a 9-bit field and is delineated thus;

D_0 & D_1: (right-most 2 bits), carry and overflow designators

D_2: (1 bit) Guard Mode enable; i.e., when this bit is a one, Guard Mode will be enabled; and when it is a zero, there is no Guard Mode.

D_3: (1 bit) storage write-only protection. Since this designator and the previous designator are somewhat interactive, their relationship may be shown by· D_3D_2=00, no guard mode and no storage protection. These two bits are automatically cleared to this state upon the occurrence of an interrupt. D_3D_2=01, Guard Mode enabled with read and write storage protection. D_3D_2=10, write-only storage protection, but no Guard Mode. (1107 mode).

D_4: (1 bit), when this bit is a one, the upper two bits are cleared from the absolute address. In this mode only the first 65K of storage can be used. This is the 1107 compatibility designator.

D_5: (1 bit). This bit controls the interrupt action when characteristic underflow occurs in double precision floating point. For f_5=0, when underflow occurs, an interrupt to location 245 is taken and no results are stored back. For f_5=1, an underflow will not cause an interrupt, but instead the result is written back as 72 zeros and the normal instruction sequence continues.

D_6: Control Register Selection Designator

D_7: Base Register Suppression Designator

D_8: 7090-94 Floating-Point Compatibility Mode

The Processor State Register is addressed only by operation code (f = 72, 15); it cannot be explicitly addressed (i.e., by the address field). The occurrence of an interrupt (any interrupt, whether I/O, arithmetic, fault, or by executing the instruction - Executive Return) is the only way the contents of PSR can be examined. While storing PSR the interrupt sequence also clears D_0 through D_5 and D_8 and sets D_6 and D_7 to one.

Description of the Basic Relative Addressing Process

Figure 5-3a, 1108 Addressing shows the sequence of operations during the formation of an address. The equivalent of five additions are performed·

1. At the start of the instruction the following two sums are formed in parallel, u + BI and u + BD. The alignment of the individual quantities is shown in Figure 5-3b, where the sums are numbered according to the keys on Figure 5-3a.
2. After the contents of the 18-bit X-register have been read out, the following three sums are formed, $u + BI + X_M$; $u + BD + X_M$ (all 3 are 18-bit adds); $u + X_M = U$.

Notice that this last addition, of the instruction u-field to the lower half of the X-register (X_M), is exactly the same as the indexing in the 1107.

The program-generated address, U, is now subjected to usual scrutiny (same as the 1107) to ascertain if it is one of the following·

1. Shift count
2. Control memory address
3. Function code<70, and j = 16 or 17 with b = 0
4. Function code<70, and j = 16 or 17 with $b \neq 0$ and i = 0
   (These last two cases are literals or u-operands.)

If one of the above four conditions exists, then everything proceeds exactly the same as in the 1107.

If, however, these conditions do not exist, then the value, U, is to be construed as a main storage address. Now the result of the comparison, U against BS (BS is the 7-bit switch register), is used to direct the appropriate address to storage. That is;

if $BS \geq U$, use $u + BI + X_M$
if $BS < U$, use $u + BD + X_M$

The comparison of BS against U is actually a 7-bit operation. This can be shown more clearly by first designating the bits of BS as; 6, . . . 0. The bits of U that are tested with BS are aligned thus

BS: 6, . . . 0
U 17, 16, 15, . . . 9.

Although there are 9-bits allotted to BS in the PSR, only the 7 lower bits are used. When loading BS, these two bits should be left zero. If they are written (by program) to other than zero, no guarantee can be given as to how the hardware will perform during the comparison. If, for example, the value 712 is loaded into BS, this value will show up as 112 when PSR is sent to (000) upon interrupt.

During the comparison, U_{16} is sampled; if it is one, the address formed with BD is automatically taken. Otherwise, when U_{16} is zero, BS_{6-0} is compared with U_{15-9} to ascertain which address is taken.

The P-Capturing Instructions

Many times during the normal course of operation, the value of P is captured when the program is temporarily halted (by an interrupt) and moved to a different location in storage (relocation). The value of P that was captured (but not yet used) while the program was resident in its old location must now reflect the new location in storage. In order to lessen the burden of programming conventions in these circumstances, necessary to insure continuity of operation of the program, the value of the captured P is automatically made program relative. The two P-capturing instructions, Store Location and Jump (f = 72, 01) and Load Modifier Jump (f = 74, 13), provide this capability. Their operation is given below.

Load Modifier Jump

1. Normal operation
   form $U + B_j$ = jump-to address
   $(P) - B_j$ = captured P (stored by an 18-bit write into $[X]_a$).
   (NOTE: The symbol B_j denotes either BI or BD).

2. Operation immediately following interrupt
   PSR → 000; clear the D field of PSR; set D_6 and D_7 to 1; see page 5-8.
   form $U + B_j$ for jump-to address (B_j now = 0)
   $(P) - B_j$ = captured P (absolute address)

Store Location and Jump

1. Normal Operation
   form $U + B_j$ for store address (of captured P)
   $U + B_j + 1$ for jump-to address
   $(P) - B_j$ = captured P (stored by an 18-bit write into $U + B_j$).

2. Operation immediately following interrupt
   PSR → 000; clear D-field. *
   form $U + B_j$ for store address (B_j now = 0)
   $U + B_j + 1$ for jump-to address
   $(P) - B_j$ = captured P

*The two operations, PSR → 000 and clear D-field as discussed on page 5-8, actually take place during the interrupt sequence before the execution of the instruction at the interrupt location. This instruction (at the interrupt location) will then operate with base registers equal to zero.

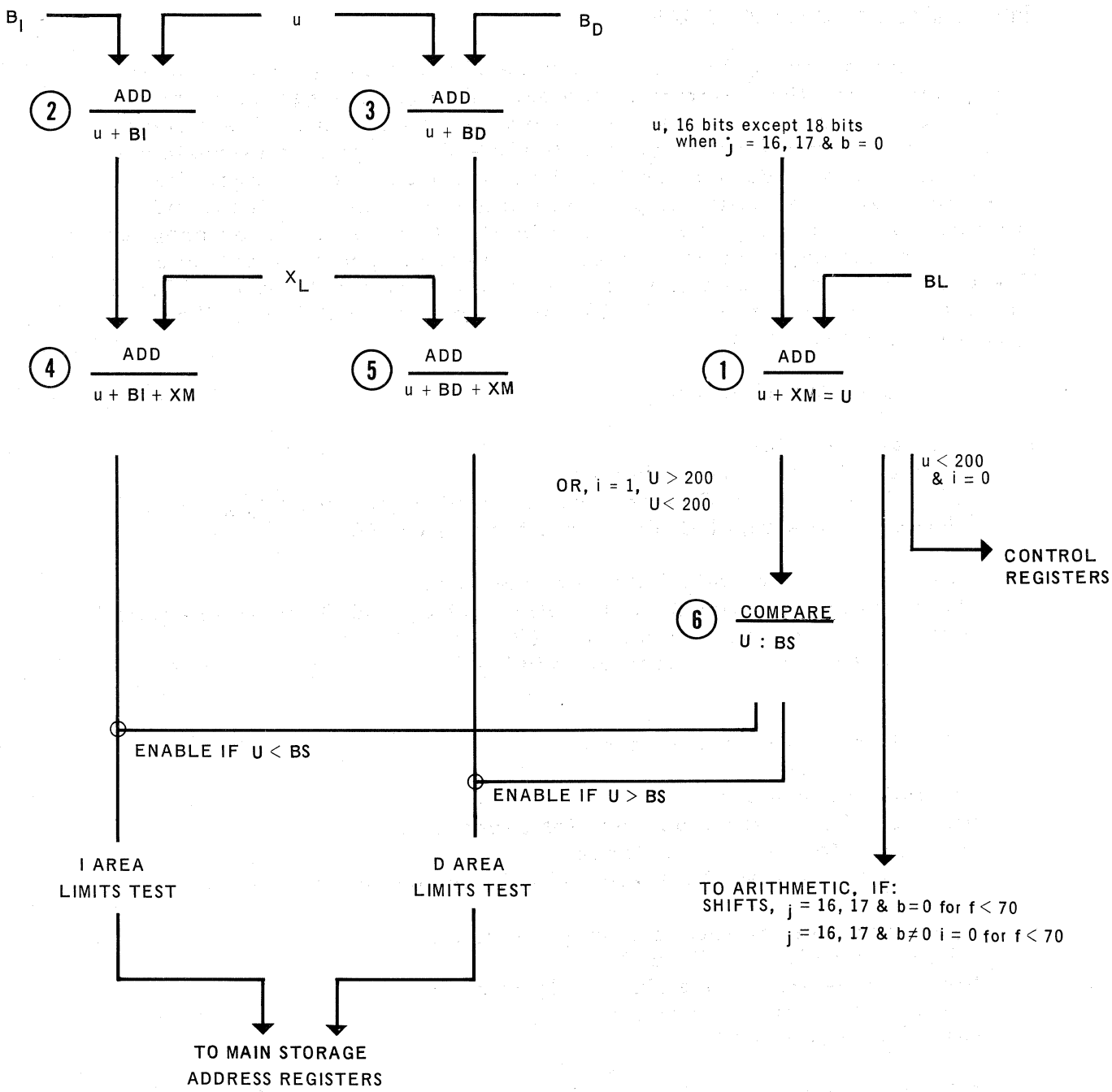
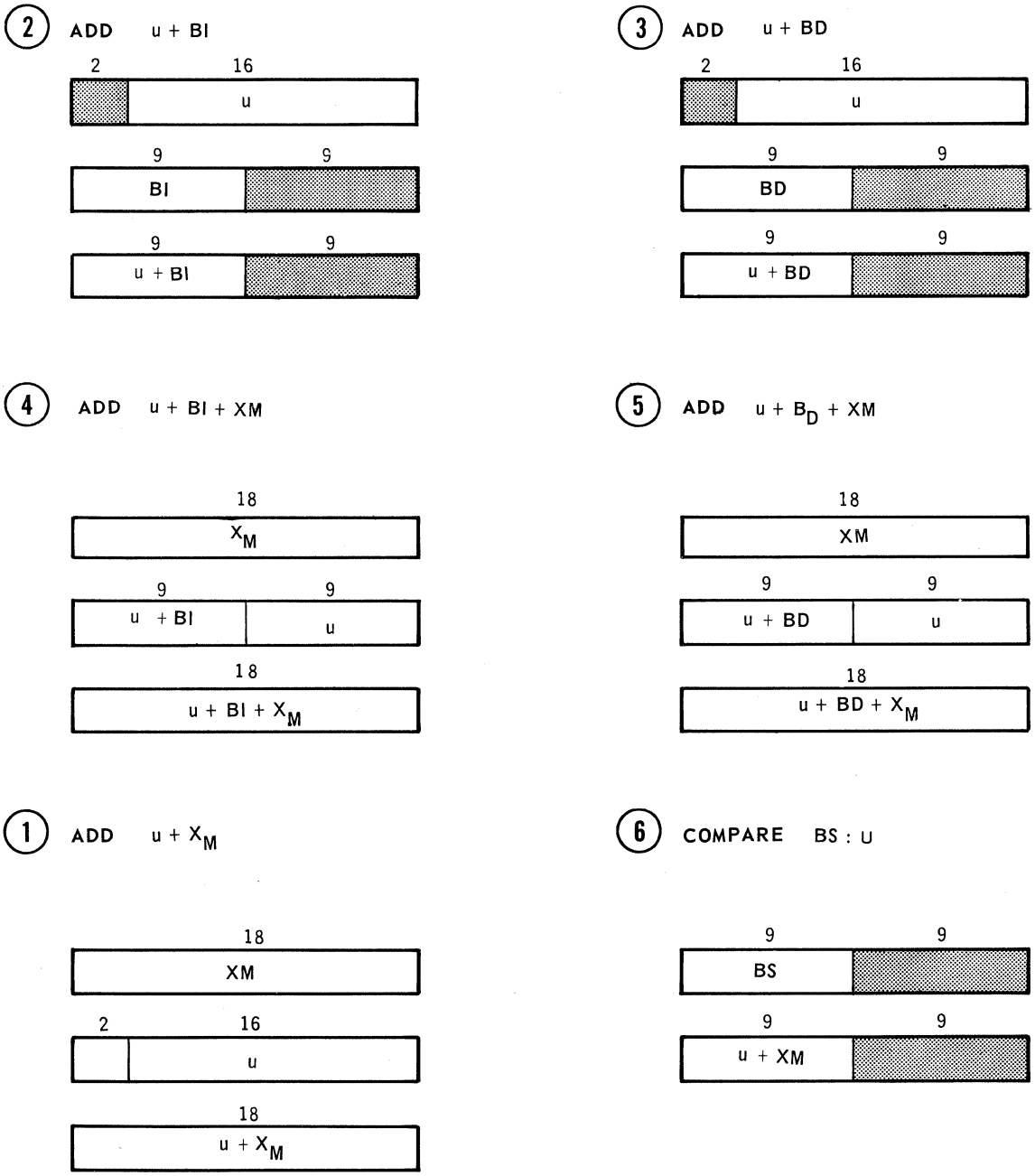


Figure 5–3a. 1108 Addressing



NOTICE THAT IN 2 AND 3 ONLY THE UPPER 7 BITS OF u ARE USED IN FORMING THE PARTIAL SUM; THE LOWER 9 BITS ARE NOT ACTUALLY USED UNTIL 4 AND 5.

Figure 5–3b. 1108 Addressing

Both of these instructions, it should be noted, incorporate the additional operation, (P) - B_j; that is, a relative, not absolute address, is captured (in normal operation - not interrupt). In a program when a jump is taken, either BI or BD will be added to the address to form the absolute, jump-to address. Which one of these two registers is added is decided on the basis of BS. This decision is remembered (in the hardware) and is used to enable, upon capturing P, the proper operation of (P) - B_j; i.e., form either (P) - BI or (P) - BD.

As an example, assume that somewhere in the course of a program a jump is taken to relative location 500 (i.e., the U of the jump instruction = 500). Now if, on the basis of the value of BS, it is determined that the address u + X_M + BI is to be used, and if BI = 10000, then the address jumped to is 10500. Now, without any further intervening jump instructions, the instruction at absolute 10507 is a Load Modifier Jump. At the initiation of this instruction, (P) = 10510; thus the value that is stored in the B-register will be (P) - BI = 510 since BI was used in forming the last P-value.

Examples showing sequence of operations upon interrupt

The following examples demonstrate the operation of relative addressing immediately following an interrupt. It is to be noted that the Return Jump or Load Modifier Jump at the interrupt location will capture the absolute address in P. This is because the interrupt sequence included the operations described on page 5-8.

Example 1: Assume the case of a working program currently at instruction address L.

L: Instruction which is interrupted, (Current value in P = L + 1)
Instruction goes to normal completion
PSR → 000 (processor state word)
Clear PSR; (see page 5-8).
Transfer control to interrupt loc. = K

K: Load Modifier (B2). Jump to 1000 (for example)
Form Jump-to address thus,
1000 + B_j (B_j now = 0)
Load into B2, (P) - B_j = L + 1 (NOTE· absolute address)

1020: (End of interrupt subroutine), Load 000 into PSR.
The new B_j in PSR and the various modes specified by PSR do not become effective until after the execution of the following instruction.
1021: Enable Interrupts and Jump to u(=0) + (B2)
In other words, jump to L + 1; (contents of PSR not yet applicable).

Example 2. Same case as above

L: Instruction which is interrupted, (P = L + 1)
Instruction goes to normal completion
PSR→ 000; see page 5-8.
Transfer control to interrupt loc. = K

K: Return Jump, to 1000
Form store address; 1000 + B_j = 1000
Store (P) - B_j = L + 1 at 1000
Form jump-to address; 1000 + B_j + 1 = 1001

Now, at the end of the interrupt subroutine, it becomes necessary to recover the 18-bit return address. This address is stored in the lower half of location 1000, but it cannot be used from there since the upper two bits of the address may not be zeros. (Specifically, in the 1108, an address greater than 65K will set the indirect-address designator to one, thus creating a spurious indirect reference.) Hence, it becomes necessary to employ some other procedure, such as first loading the address into a B-register.

Hence, at

1017: Load B2 - lower with u = 1000

1020: Load 000 into PSR.
The new B_j in PSR and the various modes specified by PSR do not become effective until after the execution of the following instruction.

1021: Enable Interrupts and Jump to u (=0) + (B2)
This effects a jump to L + 1.

Main Storage Protection

In the processor there exists the capability of specifying (by executing one instruction in the Executive routine) two areas in main storage for use by a running program. All locations in main storage, not specified as being usable by a program, are locked out to program read and write (or write-only) references. The usable portions of main storage are specifiable in gradations of 512 words. This means that an area that is 512 words long, 1024 words, 1536 words, etc. may be opened up for use.

As shown in the discussion on the use of the base registers, BI & BD, every main storage address that is generated in the course of a program is added to one of these two registers. There are two sets of storage limits registers. When an address is formed using BI, this absolute address is checked against the corresponding set of limits registers for the I area only. Similarly, addresses formed with BD

are only checked against the D area limits registers. After the final address, U_j, has been formed, it is checked against its appropriate limits thus: the reference is permitted if the following condition is met,

$$\text{Lower limit} \leq U_{j\ 17-9} < \text{upper limit}$$

Otherwise, if this condition is not satisfied, an interrupt occurs to location 243.

The format of the 36-bit storage limits register is shown below.

I Area		D Area	
9	9	9	9
upper	lower	upper	lower

The two usable main storage areas, as delineated by the two sets of limits values (one upper limit and one lower limit for each area), may be specified to be separate, partially overlapping, or the same area.

To load the 36-bit storage limits register the instruction, Load Storage Limits (f = 72, 16), is used. The operation of this instruction is, (U) → limits register; i.e., any location, as specified by the u-field of this instruction, may be referenced to load the storage limits register. Execution of this command does not enable the main storage protect feature; it only loads the storage limit values. To make main storage protection effective, the command, Load PSR (f = 72, 15), must be executed. Upon the occurrence of an interrupt, memory protection becomes inactive (PSR is cleared) and control is passed back to the Executive routine. Note, however, that neither loading the PSR nor the occurrence of an interrupt has any effect upon the values in the storage limits register; these values are only influenced by the instruction, Load Storage Limits.

The storage area excluded from program use is protected against (1), reading, writing, or jumping into; or (2), is protected against writing only. The selection of one of these two modes of protection is made by the appropriate bit setting in the PSR word. (See the description of the PSR for an explanation of how to enable storage protection).

I/O data word references are not affected by the storage protection feature. Also not checked against the limits values, is the address coming from the P-register (i.e., next sequential instruction or skip addresses; jumps, of course, are checked since the address to jump to is computed). Some precaution should be taken during assembly to preclude the possibility at the last instruction of a program of the P-register calling up the first instruction of the following program (or data). The use of the Executive Return instruction eliminates the possibility of returning into storage areas restricted by the storage limits register.

Executive Return Instruction

It is anticipated that this instruction (f = 72, 11) will be the vehicle by which a working program transfers control back to the Executive routine. The execution of this instruction performs the following operations.

1. Store PSR into 000.
2. Turn off storage protection; disable guard mode.
3. Interrupt to location 242.

Note that the u-field of this instruction is not used. However, like all instructions in the 1108, indirect addressing is operative, if so specified. Thus, if i = 1 in the instruction, a normal pass through indirect addressing will take place. Similarly, if h = 1 in this instruction, the lower half of the specified X-register will be incremented.

Example 1

At location,

L: Executive Return
This instruction transfers control to loc. 242.

242: Load Modifier (B2) Jump to u = 1000
Jump-to address is 1000 + B_j.
Stored into B2 is (P) - B_j = L + 1

At the end of the subroutine is (for example)

1020: Load 000 → PSR

1021: Enable Interrupts and Jump to u = 0 + B2
The jump-to address formed is:

$$0 + L + 1 + B_j = L + 1.$$

The newly loaded PSR becomes effective for the instruction at L + 1.

1107 Addresses on the 1108

The Processor State Register shows a bit (D_4 or PSR_{31}) reserved for the express purpose of removing the upper two bits (bits 16 and 17) from every absolute address(s) generated in the course of running a program. When the processor is running in this particular mode (i.e., D_4 = 1), the upper two bits of the 18-bit address are always written to zeros regardless of their computed state. Addresses coming from I/O access words are <u>not</u> affected in this manner; in this case, regardless of what mode the processor is in, all 18 bits of the address are used.

The generated address, $X_M + u$, destined to be used in the comparison with BS, as well as the results, $X_M + u + B_j$ lose their upper two bits. Thus, it is possible to relocate 1107 programs in the first 65K of the 1108. Note, the restriction - when in this 1107 - address mode; it is not possible to run programs (whether they are old 1107 programs, or new 1108 programs) anywhere but in the first 65K of memory.

This, of course, does not preclude the possibility of operating mixes involving 1107 (running in first 65K only) and 1108 programs (running anywhere in memory) since each program has its own private PSR settings which are operative only when that particular program is running.

The following table describes the contents of the P register when an interrupt occurs.

Table 5-1. Contents of P Register When Interrupted

Type of Interrupt	Contents of the P Register
Day Clock Interrupt	P + 1
Input Monitor Interrupt (ISI)	P + 1
Output Monitor Interrupt (ISI)	P + 1
Function Monitor Interrupt (ISI)	P + 1
External Interrupt (ISI)	P + 1
Input Monitor Interrupt (ESI)	P + 1
Output Monitor Interrupt (ESI)	P + 1
Power Loss Interrupt	P + 1
External Interrupt (ESI)	P + 1
Real-Time Clock Interrupt	P + 1
Interprocessor Interrupt #0	P + 1
Interprocessor Interrupt #1	P + 1
Storage Module 1 Parity Error	See note
Storage Module 2 Parity Error	See note
Storage Module 3 Parity Error	See note
Storage Module 4 Parity Error	See note
ICR Parity Error	See note
Illegal Instruction Interrupt	P + 1
Executive Return Interrupt	P + 1
Guard Mode Interrupt	Not Predictable
Characteristic Floating-Point Underflow Interrupt	P + 2
Characteristic Floating-Point Overflow Interrupt	P + 2
Divide Interrupt (Fixed and Floating-Point)	P + 2

Note: If parity error occurs during an I/O transfer contents is P + 1. If Error occurs in acquisition of instructions (E.G. Indirect Addressing), then not predictable.

6. INPUT/OUTPUT

GENERAL INPUT OUTPUT DESCRIPTION·

The input/output section of the UNIVAC 1108 System provides the data paths and control circuits necessary for direct communication between main storage and the peripheral equipment channels.

All references in this manual to input or output are made from the viewpoint of the processor; input is a transfer from the peripheral unit to the processor and output is a transfer from the processor to the peripheral unit.

Communications between main storage and the peripheral units of the system may occur over 8, 12, or 16 input/output channels, depending upon which option is installed. Each channel allows bidirectional transfers of data and control signals between storage and peripheral devices on that channel. Each channel has 72 data lines, 36 for input data and 35 for output data, to provide for parallel reading and writing of full 36 bit data words. In addition to the data lines, various control lines are used transmitting control signals.

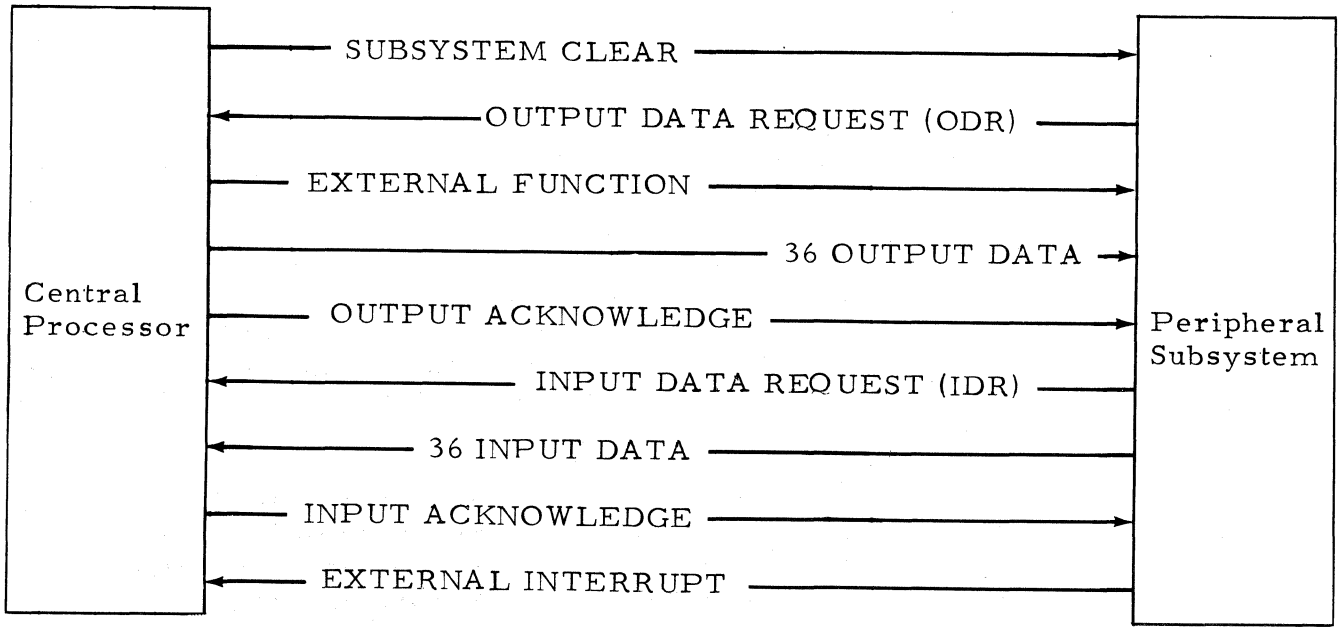


FIG. 6-1 One 1108 Input/Output Channel

The standard peripheral subsystems utilize both the input and the corresponding output lines of the same channel. Data transfers on these units (magnetic tape units, punched card units, mass storage devices, etc.) are bidirectional, but they do not occur at the same time. Data flows in only one direction over a single channel during any given interval.

Table 6-1. Input/Output Control Signals

SIGNAL	ORIGIN	DESTINATION	INTERPRETATION
Subsystem Clear	Computer	Peripheral Subsystem	All I/O channels are disengaged and clear.
External Function	Computer	Peripheral Subsystem	Indicates Function Word on the output data line.
Output Data Request	Peripheral Subsystem	Computer	Normal condition of subsystem; indicates subsystem ready for output transfer.
Output Acknowledge	Computer	Peripheral Subsystem	Indicates output word is being sent.
Input Data Request	Peripheral Subsystem	Computer	Indicates input data on data lines.
Input Acknowledge	Computer	Peripheral Subsystem	Indicates receipt of input data request and Data Word read in.

The Input/Output section functions as a small processor. Programmed Input/Output transfers load index values into the Access Control Word locations and establish the desired peripheral activity. From that point on, the I/O Control Section scans the Input Output channels automatically, accepting data from the peripheral subsystem at the natural rate of the equipment. When a peripheral subsystem transmits or requests a word of data, its associated Access Control Word is referenced as an index and the I/O Control Section transfers the data word to or from main storage, updates the Access Control Word and tests for a terminal condition.

Any one of the 1108 Input/Output Channels can be in the ISI (Internally Specified Index) mode or in the ESI (Externally Specified Index) mode. The mode of I/O channels is selected via switches on the maintenance console of the 1108. There is one switch for each channel.

PERIPHERAL SUBSYSTEMS:

Peripheral Subsystems are attached to the UNIVAC 1108 Processor through general purpose Input/Output channels. Because of the general purpose nature of these channels there is no restriction on the way that Peripheral Subsystems may be attached to them. The governing factor for peripheral attachment is the transfer rate of the devices in the subsystems. Since I/O has the highest priority and the channels are priority ordered equipment with very high transfer rates and real-time equipment is attached to the lower numbered channels which have the higher priorities.

With this adaptable Input/Output technique the UNIVAC 1108 can communicate with many real-time devices such as analog/digital converters, key sets, communication terminals, tracking and radar systems, and information processing systems.

INPUT/OUTPUT CHANNELS:

The standard UNIVAC 1108 Peripheral Subsystems are attached to the 1108 Processor by the Normal (Fast) Channels. In addition, UNIVAC 1107 Subsystems will function with the 1108 when attached by Compatible (Slow) Channels.

CHANNEL TRANSFER RATES:

The maximum data transfer rate that can be achieved on one I/O channel is:

Normal channels:	440,000 words per second (one word per 2.25 microseconds)
Compatible channels:	125,000 words per second (one word per 8 microseconds)

SYSTEM DATA TRANSFER RATES:

Normal channels:	1,333,333 words per second (one word per 750 nanoseconds)
Compatible channels:	250,000 words per second (one word per 4 microseconds)

The data rates given assume that no other processing operations are taking place and therefore can be achieved under exceptional conditions only. In more typical circumstances, concurrent program operations and the effect of functional and channel priority will reduce these rates. The actual data transfer rate achieved for a given computer configuration is a function of the types and channel assignment of the peripheral equipment and how it is used.

CHANNEL NUMBERING:

The number of I/O channels attached to a UNIVAC 1108 is optional, 8, 12, or 16 I/O channels. The assignment of channel depends upon the actual number and types of channels installed. The possible configurations are given below:

A 16-channel 1108:

$0-7_{10}$ all either Compatible or Normal

$8_{10}-15_{10}$ all either Compatible or Normal

A 12-channel 1108:

$0-7_{10}$ all either Compatible or Normal

$12_{10}-15_{10}$ all either Compatible or Normal

(With no channels numbered $8_{10}-11_{10}$)

OR, a 12-channel 1108:

$0-3_{10}$ all either Compatible or Normal

$8_{10}-15_{10}$ all either Compatible or Normal

(With no channel numbered $4_{10}-7_{10}$)

An 8-channel 1108:

$0-3_{10}$ all either Compatible or Normal

$12_{10}-15_{10}$ all either Compatible or Normal

(With no channel numbered $4-11_{10}$)

Normally, the highest-numbered channel, channel 15, is reserved for the Console. Assignment of the remaining channels will vary; however, the peripheral units with the higher transfer rates will generally be assigned the lower-numbered channels.

SUBSYSTEM ASSIGNMENT:

The peripheral devices used with the 1108 are classified as follows:

Group 1 Those which <u>must be</u> connected to a Normal Channel

Group 2 Those which <u>may be</u> connected to either a Normal or Compatible Channel

Group 3 Those which <u>must be</u> connected to a Compatible Channel

Two factors determine the classification of a peripheral device.

First, those devices with a very high transfer rate will be degraded if attached to a Compatible Channel. For example, the FH-432 Drum with an interface of one must operate on a Normal Channel. The drum can operate on a Compatible Channel but only at a much lower interface. Secondly, those devices designed for the 1107 require longer control and data signals than those presented by the Normal Channels. These units must be connected to Compatible Channels.

Group 1 Equipment:

The peripheral equipment for the 1108 which requires a Normal Channel are·

FH-432 Drum
FH-1782 Drum

The transfer time is 4 microseconds per word

Group 2 Equipment:

The following subsystems may be operated on either type of channel. For these subsystems, the transfer rate is slow enough to operate on the Compatible Channel, while the circuit and logic used in the peripheral device allows operation on a Normal Channel.

VIC/VIIIC Magnetic Tapes
Console (see note)
Intercomputer Synchronizer
FASTRAND II
Modular FASTRAND

Note: The console is a special case; because of the hardware and software restrictions; it must be connected to Channel 15. The upper group of channels ($8\text{-}15_{10}$) may be either type of channel since the console will operate with either.

Group 3 Equipment:

Peripheral devices which must operate on a Compatible Channel because of the circuit and logic limitations are listed below.

These devices currently require a Compatible Channel:

Fastrand (1107 type)
FH-880 Drum
IIIC Tape Unit
IIA Magnetic Tape Unit
M46 Printer
0751 Printer
Card Subsystem
1004 On-Site

Other peripherals which must operate on Compatible Channels:

CTS (remote 1004)
Communications Subsystem
IIIA Magnetic Tape Unit
Paper Tape Unit
Tape Adapter for UNISERVO IIIC and IVC.

INTERNALLY SPECIFIED INDEX (ISI) DATA TRANSFERS:

Each 1108 ISI Input/Output channel operates in one of three states: Input, Output, or Function State. The Input and Output States are employed when transferring data to or from the main storage. The Function State is the means by which the central processor establishes the initial communications path with a peripheral subsystem. During this state of transmission, the central processor sends one or more Function Words to a Peripheral Subsystem. These Function Words direct the units to perform a desired operation.

ISI ACCESS CONTROL WORD FORMAT:

The actual word by word transmission (regardless of the transfer state) over a given channel is governed by the I/O Control Word in the appropriate I/O register. Two of the registers, one for input and one for output, are assigned to each of the 16 channels.

In initiating an Input/Output operation, the appropriate Access Control Word is placed into one I/O Control Register by executing a Load Function in Channel, a Load Input Channel, or a Load Output Channel instruction.

Associated with these I/O Control Registers is an Input/Output Access Control Word with a fixed format. This 36 bit word consists of three designators:

G (2 bits): The incrementation designator

W (16 bits): The number of words to be transferred

V (18 bits): The initial main storage address to or from which data will be transferred.

G	W	V
35 34	33 18	17 0

The V designator portion of the ISI Access Control Word contains the address to or from which the first data word will be transferred after initiating an Input/Output operation. The W portion of the Access Control Word is tested before it is decremented by one. On input operations

if the word count W is not equal to one, the data word transfer to or from memory takes place. The V portion of the Access Control Word is increased or decreased by one according to the value of the G designator. When the word count W reaches one, the I/O transfer is terminated. The bit configurations of the G designator and the operations they specify are as follows:

G	OPERATION	NEXT ADDRESS
00	Increment V address	V + 1
01	Inhibit increment	V
10	Decrement V address	V - 1
11	Inhibit increment	V

Depending upon the value of the G designator during I/O operations, subsequent words will be transferred to or from an initial main storage address that will

1. Ascend in value, G=00
2. Descend in value, G=10
3. Remain unchanged, G=01 or 11

ISI INPUT/OUTPUT DATA TRANSFERS

The UNIVAC 1108 may have up to 16 Input/Output channels. However, regardless of the number of channels installed, each is associated with two locations in the Control Registers. One of these locations is used by an Input Access Control Word, the other one is used by an Output Access Control Word.

Control Register Addresses	USAGE
040_8-057_8	16 Input Control Registers
060_8-077_8	16 Output Control Registers

Control Registers 040_8-057_8 are the Input Control Registers assigned to channels 00_8-17_8. Control Registers 060_8-077_8 are the Output Control Registers assigned to channels 00_8-17_8. The central processor will automatically refer to these Control Registers during Input/Output operations to determine where input data is to be placed in the main storage (input data transfer) or where output data is to be taken from (output data transfer). The two Access Control Words which direct the I/O transfers on a channel are determined by the central processor, thus the name "Internally Specified Index" or ISI mode.

The ISI I/O transfers take place independently of the execution of the program instructions. Every storage cycle (750 Nanoseconds) the processor checks its pending operations to determine if the program in control should continue or if some other operations with a higher priority such as an I/O transfer should be performed. When the computer finds it necessary to perform an I/O transfer, it merely turns control over to the central processor I/O section for one storage cycle. During one storage cycle a 36 bit word or character of data can either leave or enter the computer. Immediately following this, the computer will again check and, if there is another I/O data request, it will perform the I/O data transfer because I/O has priority over initiating the next instruction. When the central processor has completed all I/O data requests present at the moment, it will access the next instruction unless the current instruction requires more time to complete execution. Each I/O data request requires a certain fixed portion of the total main storage cycle time. In the 1108, this time element for an ISI transfer is one processor cycle, 750 Nanoseconds. The 125 Nanosecond scan cycle which precedes the current I/O data transfer does not contribute to processor time because the scan cycle is performed concurrently every 125 nanoseconds with the execution of the previous instruction. ISI data transfers require one processor cycle for the actual transfer. (Steps are sometimes more convenient to think of because certain instructions require more than one 750-Nanosecond step.)

These steps are

1. A scan to detect the request (this does not add to the processor time used because of concurrence with the previous instruction operating cycle or I/O transfer time.)

2. The I/O_1 cycle (time actually taken from processor) to reference the Access Control Words and to obtain or store the data word in main storage.

3. An I/O_2 cycle to start another data transfer or execute another instruction and to send the acknowledge signal. (This does not add to the processor time because this also occurs concurrently with steps 1 and 2.)

THE VALID FUNCTION FLIP-FLOP AND ISI DATA TRANSFERS

The data transfer requires one access to main storage and, consequently, the I/O section must synchronize itself with the main storage for this purpose. The scan cycle will accomplish this by setting the Valid Function Flip-Flop if an I/O transfer is requested. This circuit enables the processor to remember that a data transfer is waiting for storage access. The processor, which is usually executing instructions, repeatedly checks this flip-flop to allow for the earliest possible chance

to make an I/O transfer. If this flip-flop has been set, then subsequent main storage references for main control are temporarily suspended until the I/O section transfers the data. If the data request was detected by the scan, the main storage access (the I/O_1 cycle) will take place at the end of the current processor cycle. The I/O section requires that the I/O scan set the Valid Function Flip-Flop at least 125 nanoseconds before the end of the processor cycle to gain access to main storage during the next processor cycle. Scanning during an I/O transfer takes place after the beginning of the I/O transfer. When scanning takes place the request with the highest priority is honored first. Once a scan has detected a request (Input, Output, External Interrupt, Day Clock, etc.), no other scanning is done. If two or more requests occur during a scan (125 nanoseconds) then priority is determined.

The Access Control Word referred to in the I/O_1 cycle is contained in the Control Registers and controls the address of the I/O data to be stored or to be extracted. Prior to initiating any I/O operation, the program must reserve an area in main storage out of which or into which data transfers will take place. They can be several hundred processor words long or as short as one processor word. The program must not only tell the processor to start the I/O transfer operation, but it must also tell the processor where the reserved areas in main storage are located. The program does this by placing the ISI Access Control Words in the appropriate ISI Control Word Locations. These locations are used by the processor to guide itself in performing the I/O data transfers.

ISI INPUT OPERATION FOR A ONE-WORD INPUT BUFFER·

For ISI input, the word count is checked before the data transfer.

Operations

1. IDR occurs.
2. Check word count W=1, therefore continue the input operation.
3. Transfer data and decrease word count.
4. Check word count again W=1.
5. Generate monitor interrupt.
6. Send ACKNOWLEDGE signal.

ISI OUTPUT OPERATION FOR A ONE-WORD OUTPUT BUFFER·

It should be noted that when dealing with an output transfer (output data or function) for a one-word buffer that the Access Control Word is checked for the zero word count before the word is obtained from storage. If W=1, the Access Control Word, word count is decreased and the data word is transferred. If W=0, the Access Control Word is not decreased, no interrupt is given, and control is returned to an ESI

transfer if one was in progress (an ISI I/O transfer can take place between the steps of an ESI transfer) or to the central processor program.

Operations

1. ODR occurs.

2. Check word count; W=1, therefore continue the output operation.

3. Transfer data and decrease word count.

4. Send ACKNOWLEDGE signal.

5. ODR occurs.

6. Check word count; W=0.

7. Abort; return to ESI I/O transfer if in progress, if not, return to central processor program.

8. Initiate monitor interrupt (if programmed).

Note that if the transferred data was a Function Word instructing the subsystem to do an input, an ODR would not follow the data transfer-an IDR would. This IDR would use a different Access Control Word and therefore, would not cause a Monitor Interrupt.

The programmer generally determines whether an instruction is to be programmed with or without a monitor interrupt. In some cases, however, the use of the monitor may be indicated by the particular operation. For example, since function transfers are essentially output transfers, the output Access Control Words control both output data and function transfers. Therefore, when programming output operations, a Load Function in Channel and Monitor (75-11) instruction would be programmed. Upon the completion of the Function Word transfer, a Monitor Interrupt will direct the program to take the next instruction from main storage location 222_8. That location will be programmed to contain a jump to start the subroutine to initiate output.

EXTERNALLY SPECIFIED INDEX (ESI) DATA TRANSFERS·

The ESI feature in conjunction with UNIVAC data communication equipment allows multiple communication lines to automatically transfer characters over a single channel to and from main storage on a self-controlled basis without disturbing the program sequence of the UNIVAC 1108 Processor. Externally Specified Index is provided for each of the sixteen Input/Output channels on the UNIVAC 1108 Processor.

Any channel may be placed in the ESI mode (except channel 15, the console channel) by actuating the appropriate switch on the maintenance panel (one switch/channel). When a switch is in the up position, the channel is in the ESI mode. When a 1108 channel is in the ESI mode,

the processor does not use the corresponding ISI Access Control Word addresses for input or output (040_8 through 077_8). The Output ISI Access Control Word addresses are used by the output Access Control Word which sends out the function words to the peripheral subsystem. The Input Access Control Word address stores the ESI Identifier. The main storage address of the ESI Access Control Word used for a particular buffer is supplied by the peripheral device or to the peripheral device with each data transfer. In this manner a large number of buffers may be active at one time. The purpose of this mode of operation is to enable the processor to transfer to and from a large number of communication peripheral devices that may be connected to a single Input/Output channel.

Many communication peripheral devices are multiplexed (combined) into one I/O channel. This requires a method to provide for as many Access Control Words as there are communication peripheral devices. To achieve this, all the requests originating from this multiplexed equipment are accompanied by an identification word. This word is used by the processor to address an area of main storage reserved for ESI Access Control Words. Each communication line has its own area in storage defined by the ESI Access Control Word. The operation wherein this identification word is given to the computer by the peripheral device is known as Externally Specified Index (ESI). It might be thought of as operating somewhat like indirect addressing, that is, the peripheral gives the processor an address (ESI) which contains the address of the actual Access Control Word for the data to be transferred.

UNIVAC 1108 COMMUNICATIONS SUBSYSTEM:

The UNIVAC 1108 Communications Subsystem enables the UNIVAC 1108 Processor to receive and transmit data via any common carrier in any of the standard rates of transmission up to 4800 bits per second. It can receive data from or transmit data to low speed, medium speed, or high speed lines in any combination.

The subsystem consists of two principle elements, the UNIVAC Communications Terminal Modules (CTM's), which make direct connection with the communication facilities, and the UNIVAC Communications Terminal Module Controller (CTMC) through which the CTM's deliver data to and receive data from the Central Processor. A Communications Terminal Module Controller may be connected to any general purpose channel. If required, a number of CTMC's may be connected through a Scanner Selector to the same general purpose channel.

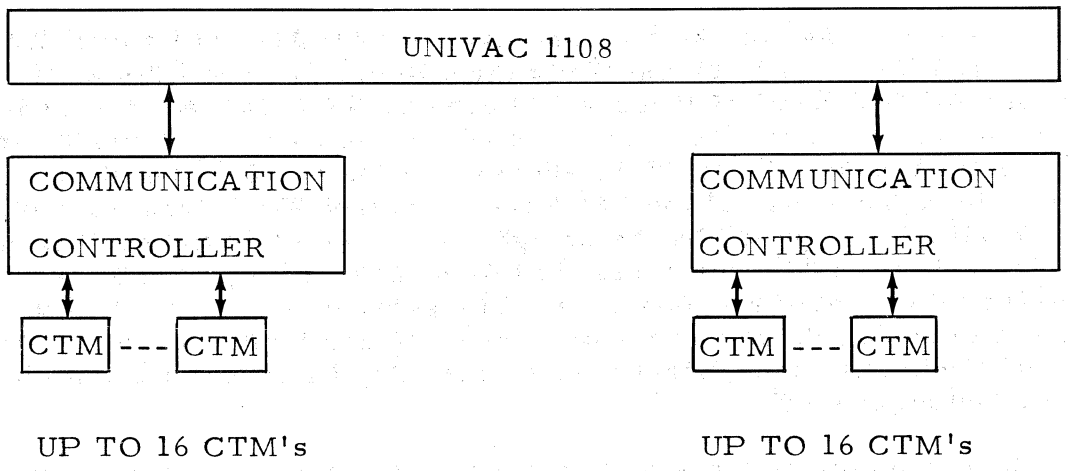


Figure 6-2 Communication Controllers Connected One per General-Purpose Channel

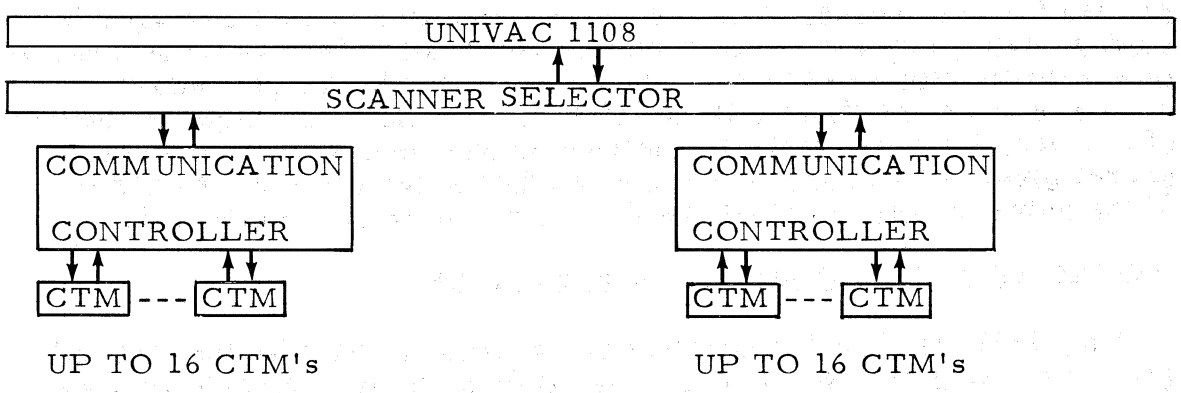


Figure 6-3 Multiple Communication Controllers Connected to General-Purpose Channel with Scanner Selector

COMMUNICATIONS TERMINAL MODULES AND COMMUNICATION LINE TERMINALS:

Table 6-2 COMMUNICATIONS TERMINAL MODULES

NAME	LOW SPEED	MEDIUM SPEED	HIGH SPEED
CODE	5,6,7,or 8 LEVEL	5,6,7,or 8 LEVEL	5,6,7,or 8 LEVEL
MODE	ASYNCHRONOUS	ASYNCHRONOUS	ASYNCHRONOUS
	BIT SERIAL	BIT SERIAL	BIT SERIAL
SPEED	UP TO 300 bps	UP TO 1600 bps	2000-4800 bps

Table 6-2 summarizes the three types of CTM's. Each type is easily adjusted to the speed and other characteristics of the type of line with which it is to operate. Each CTM accommodates two full duplex communications lines or two input and two output simplex communications lines. A CTM requires one position of the CTMC.

In addition to the CTM's, Paralleled and Dial Communications Line Terminals (CLT's) may be connected to the CTMC. The parallel CLT receives and transmits in a bit parallel rather than bit serial mode. The Dial-CLT enables the Central Processor to automatically establish communications with remote points via the common carrier's switching network. The types of input and output CLT's are summarized in Table 6-3.

Table 6-3 COMMUNICATIONS LINE TERMINALS

NAME	CLT DIALING	CLT PARALLEL
CODE	4 LEVEL	8 LEVEL
MODE	TIMING SIGNAL	TIMING SIGNAL
	BIT PARALLEL	BIT PARALLEL
SPEED	VARIABLE	UP TO 75 cps

COMMUNICATIONS TERMINAL MODULE CONTROLLER (CTMC):

The Communications Terminal Module Controller functions as a link between the processor and the CTM's and CLT's. A CTMC can handle 16 CTM's, 64 CLT's, or a mixture; this means a maximum of 32 inputs and 32 outputs can be handled by a single CTMC.

The CTM's or CLT's may request access to the main storage via the Communications Terminal Module Controller in random sequence. The CTMC automatically assigns priority among CTM's requests for access and identifies to the Central Processor the particular CTM or CLT granted access. This process is automatic and self-controlled on the UNIVAC 1108 Processor through Externally Specified Indexing (ESI) on each I/O channel.

The following types of communications service is provided

1. PRIVATE LINE TELETYPEWRITER
2. TELEX
3. TELETYPEWRITER EXCHANGE SERVICE (TWX)
4. WIDE AREA TELEPHONE SERVICE (WATS)

5. PRIVATE LINE TELEPHONE

6. DIRECT DISTANCE DIALING (DDD)

FUNCTIONAL DESCRIPTION OF COMMUNICATIONS SUBSYSTEM:

The following gives a functional description of input and output operations on the Communication Subsystem (Figure 6-4).

All odd numbered communication lines are used for input; all even numbered communication lines are used for output data transfers.

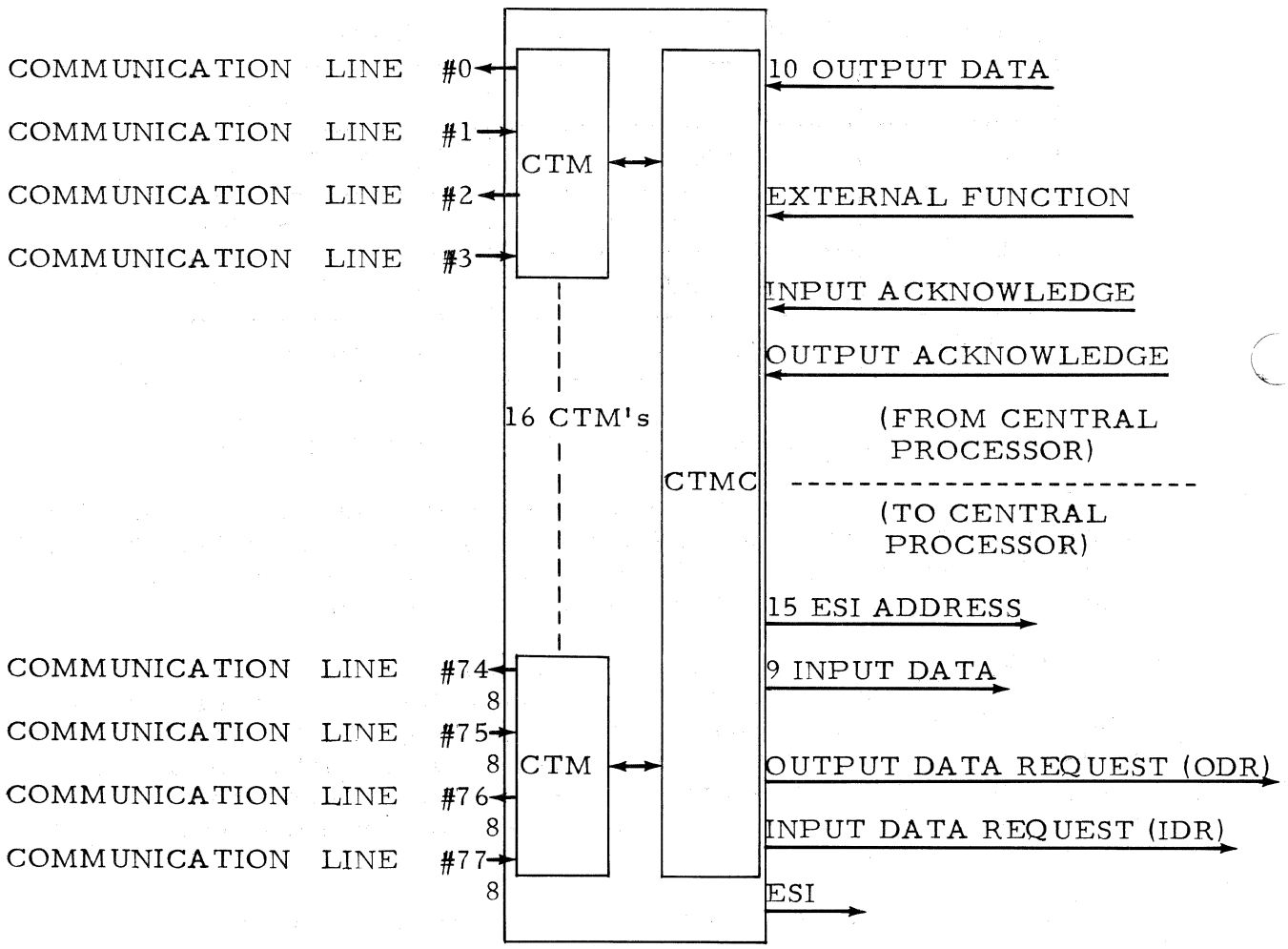


Figure 6-4 Communications Subsystem

ESI IDENTIFIER:

When communications lines transfer data characters to or from main storage via a CTMC, an ESI Identifier of 18 bits is generated as follows:

ESI BIAS	PRIORITY NUMBER
14 6	5 0

The lower 6 bits represent the Priority Number of the communications line transferring data characters. It also defines a block of 64 main storage locations which hold the ESI Access Control Words for one CTMC. Each of the 32 input and 32 output communications lines connected to a CTMC has its own ESI Access Control Word. The next 9 bits define the ESI Bias, which specifies the location of the 64 word block of ESI Access Control Words within a main storage module. Both the priority number and ESI bias are generated by the CTMC.

The 2 MSR bits are supplied from the storage select register by the Processor. They specify the main storage module which contains the ESI Access Control Words.

PRIORITY CONTROL NETWORK FOR CTMC'S:

Communication lines operations on a CTMC are sequenced and synchronized by a priority control network within the CTMC on the ESI channel. Although all communication lines on an ESI channel may be active at the same time, only one of the communication lines can use the ESI channel for a data transfer at any given time. Since several communication lines transferring data might attempt to use the ESI channel at the same time, access to the channel is given according to the following CTMC priority schedule:

Higher numbered communication lines have higher priorities than the lower numbered communication lines (see Figure 6-4). Communication line #77$_8$ has the highest priority, communication line 00$_8$ has the lower priority on a CTMC.

ESI INPUT OPERATION:

When a character is sent on an input communication line, the corresponding input on the CTMC is activated. The CTMC, which is scanning constantly, finds that this input is activated and requests the attention of the Central Processor by putting up an Input Data Request (IDR). The CTMC upon making the request presents the processor with the ESI Address via the 15 ESI address bits. When the Central Processor has completed its present operation, it honors the request from the CTMC and accepts the 15 bit ESI Access Control Word Address. The I/O section uses the ESI Address as the main part (the contents of MSR

is inserted as a high order address to form the rest of the identifier word) of the address that references the ESI Access Control Word. The ESI Access Control Word in turn gives the address where the data from this input is to be stored. Then the central processor signals the CTMC to transmit the data from this input device. The data is stored at the address designated by the ESI Access Control Word. The input operation described takes place at any time when any one of the Communication Peripheral Devices sends data.

ESI OUTPUT OPERATION:

See Appendix C, Definition of Interrupts.

ESI ACCESS CONTROL WORD AND DATA WORD FORMATS:

The format of the ESI Access Control Word is different from that of the ISI Access Control Word. This is because ESI data characters are "packed" or "unpacked" when stored into or taken out of main storage. ESI Access Control Words are stored in main storage and their location is defined by the ESI Identifier. The format is as follows:

G (2 bits): The incrementation designator

H (1 bit): The half word designator

W (15 bits): Character count of message

V (18 bits): The initial main storage address to or from which data will be transferred.

G	H	W	V
35 34	33	32 18	17 0

The V designator portion of the ESI Access Control Word contains the address to or from which the first data character will be transferred. The V address portion will be increased or decreased according to the G designator - once every two characters; this is conditional on the state of H. When

H = 0, use the lower half of the location specified by V, do not increment or decrement V according to G.

H = 1, use the upper half of the location specified by V, increment or decrement V according to G.

The G designator bit configurations and operations they specify are as follows:

G	OPERATION	NEXT ADDRESS
00	Increment V address if H = 1	V + 1
01	Inhibit increment	V
10	Decrement V address if H = 1	V - 1
11	Inhibit increment	V

The format of the ESI data word is

CHARACTER 35 — 18	CHARACTER 17 — 0

The first character of an incoming message or a message being sent out will cause the appropriate ESI Access Control Word to be read out of the ESI Access Control Word main storage location. If H = 0, then the data character will be transferred to or from the lower half of the location specified by V. Regardless of the setting of G, V will not be altered. The H bit is now set to one by the hardware, W is decremented by one, and the control word is stored back. The second character of the same message transferred will cause the ESI Access Control Word to be read out again. Since H = 1, this time the second data character will be transferred into or from the upper half of location V. This time since H = 1, V will be modified according to G. H is automatically set back to zero, W is decremented, and the process repeated for all following data character transfers.

ESI INPUT/OUTPUT DATA TRANSFERS:

Due to the fact that whenever an ESI data transfer takes place the ESI Address and the MSR give the main storage location of the corresponding ESI Access Control Word, the ESI transfers require more time than data transfers in the ISI mode. The actual ESI data transfer requires two or three processor cycles. This depends on whether the data and the Access Control Words are stored in the same or different modules of main storage or if the Access Control Words are not in Module 1.

To illustrate, there are four or five (steps) associated with an ESI data transfer.

The steps (processor cycles) associated with an ESI data transfer are:

STEP 1. A scan to detect the request (this does not add to the total time required because the 125 nanosecond scan is done concurrently with the instruction cycle or the data transfer).

STEP 2. The I/O_1 cycle to Read the ESI Access Control Word from main storage. *

STEP 3. The I/O_2 cycle to write the ESI Access Control Word in main storage. If the ESI Access Control Words are in Module 1 and data is not in the same main storage module--then step 3 and step 4 are done at the same time.

STEP 4. The I/O_3 cycle obtains or stores the data word in main storage. *

Data transfers using ESI usually require two "step-times" even though three steps are actually done (step 2 and steps 3 & 4). This is because steps 3 & 4 can be done concurrently. But, if the Access Control Word and the data to be transferred are in the same main storage module or if the Access Control Words are not in main storage module 1, three "step-times" are automatically required to do the ESI transfer (2.25 microseconds). Note that ISI I/O transfers can also take place between the main storage reference steps of the ESI transfer.

The ESI Access Control Words do not operate within the Control Registers, but in main storage; therefore, ESI data transfers take longer than the ISI counterparts.

THE VALID FUNCTION FLIP-FLOP AND ESI DATA TRANSFERS:

The control of the Valid Function Flip-Flop is the same for an ESI data transfer as for an ISI data transfer.

ESI INPUT OPERATION FOR A ONE-WORD INPUT BUFFER:

The following events occur for an ESI data input transfer:

Operations

1. IDR occurs.

2. Check character count; W = 1; therefore go on with transfer.

3. Transfer data and decrease character count.

---

*This is time taken away from the execution of the main program's instructions.

4. Check character count; W = 0.

5. Initiate Monitor.

6. Send ACKNOWLEDGE.

   Note: If the channel is activated, but the character count is zero (W = 0), the following operation occurs.

7. IDR occurs on the peripheral device.

8. Check character count, and if W = 0, no data is transferred; Access Control Word is not changed.

9. Send ACKNOWLEDGE.

ESI OUTPUT OPERATION FOR A ONE-WORD OUTPUT BUFFER:

The following events occur for a one-word output buffer.

1. ODR occurs.

2. Check character count, W = 1.

3. Transfer data and decrease character count.

4. Check character count, W = 0.

5. Initiate monitor.

6. Initiate ACKNOWLEDGE signal.

   Note: If the channel is activated, but the character count is zero (W = 0), the following operations occur.

7. ODR occurs on the peripheral device.

8. Check character count; W = 0. Transfer all ones; Access Control Word is not changed.

9. Send ACKNOWLEDGE.

MAIN STORAGE ACCESS CONTROL OF I/O OPERATIONS:

Channel operations are sequenced and synchronized by a Priority Control Network within the Input/Output Section of the UNIVAC 1108 Processor.

Although all 16 channels may have data transmissions pending between main storage and peripheral units at the same time, at most, only one channel can actually make a data transfer to main storage during each 750 nanosecond main storage cycle.

Priority control circuits, along with the access control circuits, resolve these situations where two or more channels simultaneously attempt to communicate with the Processor. Access to the UNIVAC 1108 Processor is granted on the basis of a Channel Priority Control Network.

INPUT/OUTPUT INTERRUPTS:

The types of Input/Output Interrupts and their causes are explained in detail in Appendix C.

INTERRUPT PRIORITY:

The Interrupt Priority Control Network is explained in detail in the PRIORITY CONTROL NETWORK Chapter.

PROGRAMMING CONSIDERATIONS FOR I/O INTERRUPTS:

When I/O Interrupts occur the hardware defined sequence of events takes place as explained in detail in the Appendix C.

The occurrence of an External Interrupt on an ISI or ESI channel causes a Status Word to be stored into the External Status Word location 230_8. The Status Word contains information which was generated by the Peripheral Control Unit and the channel synchronizer. External Interrupts can be requested by the program to indicate normal completion of an operation.

The Status Word is always generated by the subsystem when it sends the External Interrupt, i.e., when the External Interrupt becomes effective, the Status Word will be in main storage location 230_8 at that time. Since the occurrence of an External Interrupt disables all other I/O Interrupts, including all other External Interrupts, the corresponding Status Words to the other External Interrupts will not be lost but simply delayed until the other External Interrupts become effective.

The Status Word is sent to the central processor in the same manner as the input data words, except that an External Interrupt signal is generated after the Channel Synchronizer has placed the Status Word on the input data lines. In this way the central processor can distinguish Status Words from input data words.

THE ISI STATUS WORD FORMAT:

The ISI Status Word is a 36 bit word and is transmitted over the input data lines to main storage locations 230_8. The format of the Status Word depends upon the type of Peripheral Subsystem that generated the External Interrupt. For detailed information concerning the format of the Status Words generated by a particular subsystem see the hardware description of that subsystem. In general the Status Word specifies the type of error and identifies the unit of the subsystem where the error was detected. If an I/O operation was programmed which asked for the External Interrupt upon normal completion of the I/O operation (see Appendix C), then the contents of the Status Word will indicate this fact.

THE ESI STATUS WORD FORMAT:

At present none of the communication devices that can be attached to an ESI channel generate External Interrupts and; therefore, no ESI Status Words are generated. This information will be supplied later when applicable.

PROGRAMMING THE INPUT/OUTPUT OPERATIONS:

The stored program of instructions directs the 1108 computer's input/output section. The program activates a particular mode (function, input or output) on the I/O channel by means of the Load Input Channel (75-00), the Load Input Channel and Monitor (75-01), the Load Output Channel and Monitor (75-05), the Load Function In Channel Instruction (75-10), or the Load Function in Channel and Monitor (75-11) Instructions.

These instructions activate the particular mode (function, input or output) for the channel and store the appropriate Access Control Words necessary for controlling the transfers of information to and from the computer.

Each of the I/O channels is operated separately after it is activated; that is, the channel operates automatically and independently of the main computer program until the operation is complete or terminated.

THE I/O INSTRUCTION WORD FORMAT:

The basic I/O Instruction Word consists of seven sections (fields) each represented by a letter.

FUNCTION		CHAN. MOD.	OPERAND			
f 35 30	j 29 26	a 25 22	b 21 18	h 17	i 16	u 15 00

During the execution of an I/O instruction, the j-field does not serve as a partial-word determinant as it does in other instructions, but combines with the f-field to form a 10-bit function code. The a-field specifies the I/O channel, if one is needed for the operation (see CSR Register). Two I/O Access Control Words are associated with each channel (one for input and one for output). The control register address to which the referenced I/O Access Control Word is to be transferred when the instruction is executed is specified "indirectly" via the channel number by the a-field. (The b-, h-, and i-fields determine Indexing Modification, Incrementation, and Indirect Addressing, respectively according to the normal rules.) While the u-field (after being modified by Indexing and Relative Addressing) specifies the address in the main storage containing the I/O Access Control Word.

If the Channel Select Register (CSR) is loaded by a Load Channel Select Register (73-16) instruction, all I/O instructions will use the logical add (⊕) of the a-field and the contents of the CSR Register to specify the channel number.

When the I/O Instruction Word is decoded, it gives the mode of operation, activates the selected channel, initiates the transfer, loads the Access Control Word into its respective I/O Control Register, and conditions the computer I/O section for the transfer. Once the mode is established and the transfer initiated, the input/output section continues independently of the main program (but the main program is stopped during the actual I/O transfer one main storage cycle time for ISI data transfers). From the time of initiation, the data transfer is controlled by the Access Control Word. The W-Portion of the word acts as a counter, is modified according to G, and is tested for zero as each Data Word is transferred either into or out of the main storage. If no error or malfunction occurs, the data transfer will continue until the count reaches zero, or until in the case of input, the peripheral subsystem stops sending data. When the count is zero, a signal is generated to end the I/O transfer sequence, and a monitor interrupt is generated (if programmed).

THE FUNCTION WORD FORMAT:

Function Words contain the operating commands for the peripheral subsystems. They indicate what type of operation is to be performed and which unit of the subsystem is to be used. Function Words are sent to the subsystem by programming the Load Function in Channel (75-10) or Load Function in Channel and Monitor (75-11) instructions. Each Function Word is accompanied by an External Function signal (see Figure 6-1 and Table 6-1) from the central processor when sent to the subsystem. The External Function signal serves to differentiate, in the Channel Synchronizer, the Function Word from a Data Word.

ISI FUNCTION WORD FORMAT:

The format of ISI Function Words depends upon the type of Peripheral Subsystem addressed. For details concerning the format of a Function Word for a particular Subsystem refer to the hardware description of the Subsystem. Function Words are 36 bit words long.

ESI FUNCTION WORD FORMAT:

ESI Function Words are used for programming CTMC's. The format of ESI Function Words is as follows:

NOT USED	CTMC	DEVICE NUMBER	FUNCTION CODE
35 14	13 10	9 3	2 0

Function Code = 3-bit Function Code

Line Number = 7-bit Line identifier code (Tells for which line the Function Code is being sent).

CTMC = 4-bit CTMC code (More than one CTMC can be on a channel. The CTMC code is used to designate to which CTMC the function is to go.)

THE DATA WORD FORMATS:

The format of the Data Words to be transferred during input/output operations varies depending upon ISI or ESI data transfers.

ISI DATA WORD FORMAT:

Data words may be any 36 bit configuration representing the data to be transferred during ISI I/O transfers. Data Words may be either numeric or alphanumeric in a binary or any binary coded configuration.

Peripheral Subsystems attached directly to an ISI channel without a Channel Synchronizer transfer characters only. The characters of variable bit length, depending upon the type of subsystem, may be numeric or alphanumeric in a binary or any binary coded configuration.

ESI DATA WORD FORMAT:

ESI channel I/O transfers transmit information in character format only. The data characters of variable bit length, depending upon the type of communication device used, may be numeric or alphanumeric in a binary or any binary coded configuration are imbedded in half words.

THE IDENTIFIER WORD:

Identifier words are used in conjunction with Search Operation. The word may have any 36-bit binary composition. No format is specified. During a Search Read Function, for example, the Identifier Word is held in a Channel Synchronizer register. As a tape or drum is searched, the Identifier Word is compared to the first word read of each block. When a match occurs, all or part of the block is read into the main storage.

THE MASK WORD:

During a mask operation, the Mask Word validates those bits of the Identifier Word to which comparison is made. No format is specified. It may be any 36-bit binary composition. A one in the Mask Word causes the corresponding bit of the Identifier Word to be compared to the word called up for search comparison. A zero masks out the corresponding bit.

Identifier Word	1 0 1 1 0 1 1 0 1 1
	↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
Mask Word	1 1 1 1 1 0 0 0 0 0
Identifier Word	↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
(After Mask)	1 0 1 1 0 0 0 0 0 0

masked out

PROGRAMMING AN ISI OUTPUT DATA TRANSFER:

In programming instructions for an output data transfer, the following sequence must be observed:

1. Program Function Mode

2. Program Output Mode

This sequence conditions the channel for the desired function and initiates the data transfer.

The sequence

1. Program Output Mode

2. Program Function Mode

does not work, because the Output Access Control Word stored into the Output Access Control Word location by the Load Output Channel instructions would be destroyed by the Output Access Control Word stored into the same location when executing the Load Function in Channel instructions.

FUNCTION MODE:

When the Load Function In Channel (75-11) or Load Function In Channel and Monitor (75-11) instruction is executed, the "U"-referenced Access Control Word for the selected channel is transferred to its respective I/O Control Register, e.g., channel number 1 uses CR location 061_8 for output and loads the ACN from the effective address "U". The output channel is now activated (the Access Control Words sent because of Load Function in Channel instructions are considered to be Output Access Control Words).

The Function Word which is at the address given by the V-Portion of the Access Control Word is sent to the peripheral subsystem. The computer sends an External Function signal to the Channel Synchronizer indicating that a Function Word is on the data lines. Some functions may require more than one Function Word. This is indicated by the word count in the W-Portion of the Access Control Word.

The Function Word is decoded in the Channel Synchronizer to select a unit on this channel and to specify the operation to be performed by this unit. When the selected unit is ready, an Output Data Request is generated by the subsystem and returned to the computer.

OUTPUT MODE

Having been conditioned for an output operation, the subsystem is ready to receive data from the computer. To know how many words are to be transferred and at what address the initial transfer is to begin, the Load Output Channel (75-04) or Load Output Channel and Monitor (75-05) instruction is executed.

At this time the Access Control Word for the channel is transferred to its respective Control Register address (Channel Number 1 uses the control register address 061_8 for Output) as derived from the effective address U of the output channel instruction. The Output Mode on this channel is now activated.

The Access Control Word sent by this instruction provides the information for the address of the data transfer (V portion) and how many words to transfer (W portion). The first Data Word is placed on the data lines and the transfer operation starts.

With output data transfers, the same control register is used to contain the Access Control Words of both the Load Function and the Load Output instructions. Since the Output Access Control Word for the data transfer is the one that must be referenced last and for the longer time, it must follow the output Access Control Word sent out by the Load Function in Channel instruction. To prevent the Output Access Control Words from destroying each other, programmers may use the Jump On Function In Channel instruction (75-12). This instruction effects a delay and notifies the computer when the function mode has been terminated and that the computer may activate the output mode.

This control register sharing is not used for input data transfers, therefore, the order of operation is not dependent on the computer, but on the peripheral device.

PROGRAMMING AN ISI INPUT DATA TRANSFER:

In programming instructions for an input data transfer from the peripheral subsystem to the computer, the following sequence must be observed:

1. Program Input Mode

2. Program Function Mode

Programming this sequence can cause problems since interrupts, IDR's, and ODR's of other channels, which might occur right after the Function Mode was initiated, can delay the activating of the Input Mode so that the subsystem tries to send input data to the Central Processor before the channel is in the Input Mode. This sequence, therefore, can result in a loss of input data.

INPUT MODE:

By programming the Load Input Channel (75-00) or Load Input Channel and Monitor (75-01) instruction the Input Access Control Word is transferred to its respective Control Register address (Channel Number 1 uses the Access Control Word at Control Register address 41_8 for input) as derived from the effective address U. The Input Mode on this channel is now activated.

PROGRAMMING AN ESI INPUT DATA TRANSFER:

The following discussion will explain how ESI Input Data Transfers are programmed on the UNIVAC 1108 Communications Subsystem. ESI will work with other specially designed devices also.

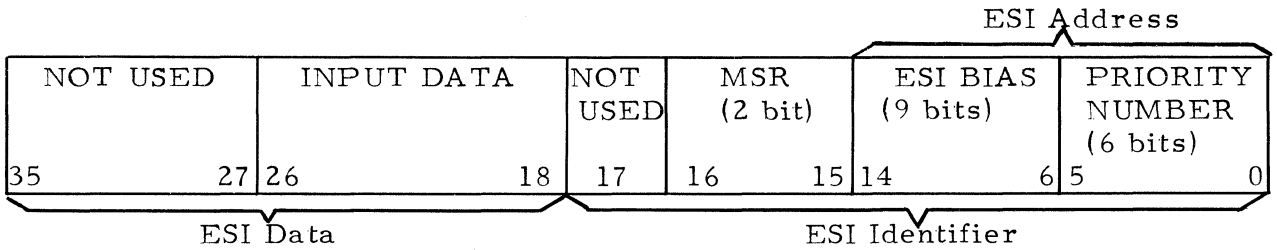
Before input data can be transferred to the computer, the program must complete the following steps:

STEP 1. Load the main storage address that will be specified by the Externally Specified Index (ESI) with the ESI Access Control Words.

STEP 2. Activate the channel to be used for the data transfer.

STEP 3. Send the proper function to the Communications Subsystem. This step is required to activate input data transfers for high speed CTM's only.

When a communications peripheral device is ready to transmit a Data Word to the computer via the Communications Subsystem, an ESI address in the lower 15-bits of the 36-bit input word is sent along with the upper 18-bits where the ESI Data is located. Together, the two parts, ESI address (15 bits) and the ESI Data (18 bits), make up one input word as shown below.



During input operation, the ESI Identifier (18 bits) are stored in the upper 18 bits of the ISI Input Access Control Word location. (With channel 0 using 40_8, channel 1 using 41_8, etc.) In addition to storing the ESI Identifier in the ISI Input Access Control Word location, the ESI Identifier is decoded by the I/O hardware.

The MSR part of the ESI Identifier indicates in which main storage module the ESI Access Control Words are located.

The ESI Bias tells where (in the selected main storage module) the block of 64 ESI Access Control Words of a CTMC are located.

The Priority Number tells which particular word of the 64 word block of ESI Access Control Words is the one for the requesting input device.

The following explains the three steps that must be programmed initially for ESI Input Data transfers in more detail.

STEP 1. These 64 core main storage locations must be loaded with the ESI Access Control Words by the programmer before an ESI input operation can start. Therefore, the programmer must know the ESI Bias for the Communications Subsystem and load the ESI Access Control Words accordingly. The MSR portion of the ESI identifier will be supplied by the central processor.

STEP 2. To activate the channel for a data transfer, the Load Input Channel and Monitor instruction (75-01) should be used. This instruction works differently in the ESI mode. When in the ESI mode, the instruction enables the I/O section to look for Input Data Requests, and, although it does attempt to transfer the contents of U to the appropriate Input ISI Access Control Word Address, the transfer is inhibited and nothing is transferred into the Input ISI Access Control Word Address. To speed up this inhibited transfer, it is beneficial to designate U with an address less than 200_8 from a timing standpoint. The Load Input Channel and Monitor instruction (75-01) should be used rather than the normal Load Input Channel (75-00) because a check is needed to tell the computer when the buffer has been completely filled.

STEP 3. For low-speed and medium-speed inputs on CTM's, a Function Word is not required to start input, however, for high speed CTM's a Function Word is required.

PROGRAMMING AN ESI OUTPUT DATA TRANSFER:

The following discussion will explain how ESI Output Data Transfers are programmed on the 1108 Communications Subsystem. ESI will work with other specially designed devices also.

Before an output data transfer can take place, the program must perform the following general steps:

STEP 1. Load the ESI Access Control Word into address specified by the Externally Specified Index.

STEP 2. Activate the channel to be used for the data transfer.

STEP 3. Send the proper Function Word to the Communication Subsystem. This step is required to activate output data transfers for all outputs on CTM's.

The previously listed general steps are given below in more detail.

STEP 1. This is done exactly the same as for step 1 in ESI Input.

STEP 2. Before an output can take place, the channel must be activated. To activate the channel for an ESI data transfer, the Load Output Channel and Monitor (75-05) will be used.

When in the ESI Mode this instruction enables the computer to honor Output Data Requests. This instruction attempts to transfer the contents of the U-address to the appropriate ISI Output Access Control Address in the control registers, but the data is not written into them - it is inhibited. To speed up the execute time of this instruction the U-address should be equal to a number less than 200_8. Therefore, the attempted transfer would be from one control register to another. After this instruction has actuated the channel, the 1108 will honor ODR's from the Communications Subsystem.

For all outputs on CTM's, a function word is required.

STEP 3. A Function Word must be sent to the Communication Subsystem. The word will be in the standard I/O instruction word format. It will request the Communication Subsystem to return to the computer an ODR and ESI Identifier. The computer will recognize them because the channel activity has now been initiated. The ESI Identifier will determine the locations of the ESI Access Control Word. The ESI Identifier will be stored in the lower 18-bits of the ISI Input Access Control Address (040_8-057_8 with Channel 0 using address 40_8, Channel 1 using 41_8, etc.). The reason this ESI Identifier is stored in the ISI Input Access Control Word Location is that in case the ESI transfer is interrupted the program will only have to examine the Identifier to determine on which channel and output device the last transfer was done.

PROGRAMMING UNIVAC 1108 PERIPHERAL SUBSYSTEMS (ISI):

To understand how to program Output as well as Input with the 1108 peripheral devices, the following examples are given:

The 75-06 (JOC) Jump on Output Channel Busy is the only test instruction that depends on a reaction from the peripheral subsystem and, therefore, is the only instruction requiring a timeout (a clock counting the time required to establish the mode). This is to make sure the computer does not wait unduly long for the ODR. The jump on Input Channel Busy Instruction would require timeout if the word count was not checked after the transfer as well as before.

Two types of computer words also are used for Input Data Transfers; they are the Identifier Word and the Mask Word.

MAGNETIC TAPES, FASTRAND, HIGH SPEED PRINTER SUBSYSTEMS:

The VIC and VIIC Magnetic Tape Subsystems, the 0751 High-Speed Printer Subsystem, and Fastrand Subsystem operate in the same fashion for input and output with the 1108. To program the I/O section for any one of the devices we could do the following:

INPUT:

75-00 (LIC) Load Input Channel (initiate input).
(do other work)

75-10 (LFC) Load Function in Channel (1-word function buffer).
(do other work)

75-12 (JFC) Jump on Function in Channel (test function leaving computer).

OUTPUT:

75-10 (LFC) Load Function in Channel (1-word function buffer).
(do other work)

75-12 (JFC) Jump on Function in Channel (test function leaving computer).

75-04 (LOC) Load Output Channel (initiate output buffer).

THE FH-432 AND THE FH-880 DRUM SUBSYSTEMS:

The FH-432 and the FH-880 Drum Subsystems require different programming than the Tapes, Fastrand, and High-Speed Printer. They require that a programmer working with the drum subsystem use the Drum's Terminate Instruction and then proceed with establishing the next function whether it be input or output.

DRUM INPUT:

75-10 (LFC) Load Function in Channel (1-word buffer containing terminate function). (do other work)

75-06 (JOC) Jump on Output Channel Busy (test function output and wait for ODR).

75-00 (LIC) Load Input Channel (activate input buffer).

75-10 (LFC) Load Function in Channel (1-word buffer containing function word). (do other work)

Note that between the I/O instruction of the following programs, other instructions may be executed.

DRUM OUTPUT:

75-10 (LFC) Load Function in Channel (1-word buffer containing terminate function).

75-06 (JOC) Jump on Output Channel Busy (test function transfer and wait for ODR).

75-10 (LFC) Load Function in Channel (1-word buffer containing write function).

75-12 (JFC) Jump on Function in Channel (wait until function leaves computer).

75-05 (LOCM) Load Output Channel and Monitor (activate output and monitor).

DRUM SEARCH READ:

75-10 (LFC) Load Function in Channel (1-word buffer containing terminate function)

75-06 (JOC) Jump on Output Channel Busy (test function transfer and wait for ODR).

75-00 (LIC) Load Input Channel (activate input buffer).

75-10 (LFC) Load Function in Channel (2-word function buffer with search function and Identifier Word).

DRUM (DUAL COMPUTER) INPUT:

75-10 (LFC) Load Function in Channel (1-word buffer containing terminate function).

75-06 (JOC) Jump on Output Channel (test function transfer and wait for ODR).

75-00 (LIC) Load Input Channel (activate input).

75-10 (LFC) Load Function in Channel (2-word buffer containing release control function and read function).

DRUM (DUAL COMPUTER) OUTPUT:

75-10 (LFC) Load Function in Channel (1-word buffer containing terminate function).

75-06 (JOC) Jump on Output Channel Busy (test function transfer and wait for ODR).

75-10 (LFC) Load Function in Channel (1-word buffer with release control function).

75-12 (JFC) Jump on Function in Channel (wait for function to leave computer).

75-10 (LFC) Load Function in Channel (1-word function buffer with output function).

75-12 (JFC) Jump on Function in Channel (function has left the computer).

75-05 (LOCM) Load Output Channel and Monitor (output buffer and monitor).

DRUM (DUAL COMPUTER) INPUT (A SECOND METHOD):

75-10 (LFC) Load Function in Channel (1-word buffer containing terminate function).

75-06 (JOC) Jump on Output Channel (test function transfer and wait for ODR).

75-01 (LICM) Load Input Channel and Monitor (activate input buffer and monitor).

75-10 (LFC) Load Function in Channel (1-word containing release control).

75-12 (JOC) Jump on Function in Channel (wait until function leaves computer).

75-10 (LFC) Load Function in Channel (1-word containing Function Word).

DRUM (DUAL COMPUTER) SEARCH READ:

75-10 (LFC) Load Function in Channel (1-word buffer containing terminate function).

75-06 (JOC) Jump on Output Channel (test function transfer and wait for ODR).

75-00 (LIC) Load Input Channel (activate input buffer).

75-10 (LFC) Load Function in Channel (1-word buffer contains release control).

75-12 (JFC) Jump on Function in Channel (wait until function leaves computer).

75-10 (LFC) Load Function in Channel (2-word buffer with search function and Identifier Word).

PROGRAMMING THE COMMUNICATION SUBSYSTEM (ESI):

Because the Communications Subsystem is the only device presently planned to use ESI on the 1108 Processor and due to the relative complexity of the ESI, the following examples will be given as a method of programming. It must be remembered that these are only examples to help the programmer get started.

ESI INPUT:

The chain of events for input data transfers which operates independently of the program is:

The input device presents a Service Request Signal to the CTMC upon the receipt of a full Data Word (character) from the communication peripheral device. The CTMC then "locks on" to this device and sends the Input Data Request and ESI Identifier to the computer. The computer, upon receipt of the Input Data Request and ESI Identifier, gets ready to receive the data in its already established ESI buffer. When the computer has received data from the CTM, it sends an Input Acknowledge. The CTM drops its service request signal upon receiving the Input Acknowledge from the computer. This process is repeated each time the input receives valid data from the communication peripheral device.

The CLT operates in the same fashion as the other CTM devices except that it transmits continously whether there is valid data or not. Therefore an External Function is sent to the CLT for synchronization purposes. This terminates the input process.

ESI INITIAL LOAD INPUT:

05-XX (SZ) Store zero (zero all ESI addresses)

10-XX (LA) Load A (load the ESI Access Control Word in Register A).

01-XX (SA) Store A (Store the ESI Access Control Word in ESI Address)

75-01 (LICM) Load Input Channel and Monitor (causes the computer to honor IDR's; set U less than 200_8 in order to speed up the execution time of this instruction).

This short program (Figure 6-5) establishes and initiates all the buffers when the computer is turned on. After the computer is in operation, the I/O section transfers the data independently of the program. The next time the program becomes involved with the ESI and the Communications Subsystem is when a buffer has been completely filled. Then a Monitor Interrupt is generated to the program and the program honors it according to the block diagram in Figure 6-6.

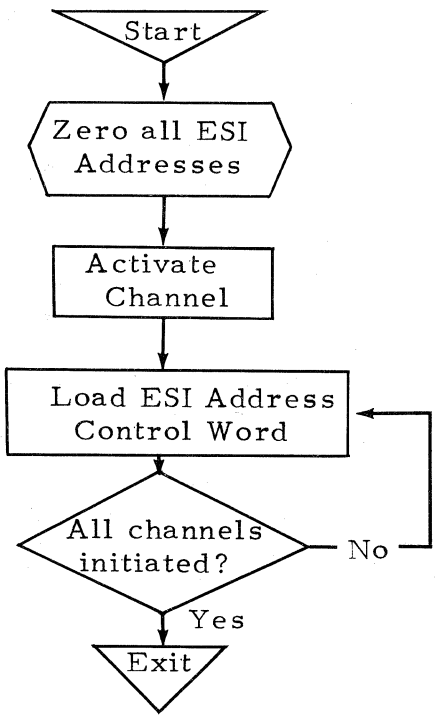


Figure 6-5 Flow Chart of ESI Initial Load Input Subroutine

PROGRAMMING ESI INPUT:

When the ESI Monitor Interrupt causes the program to go to its assigned Interrupt address, the interrupt handling subroutine (Figure 6-6) is initiated.

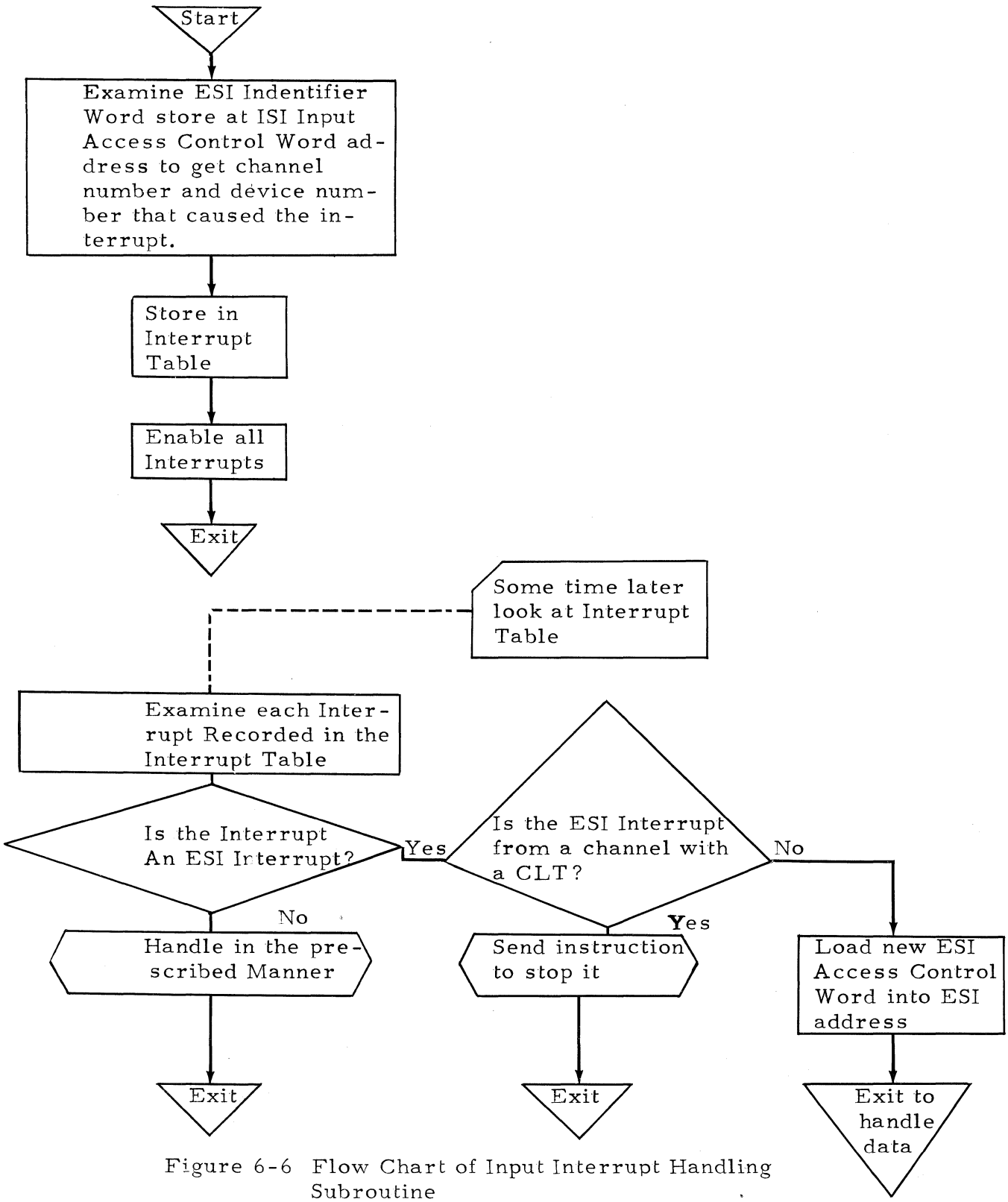


Figure 6-6 Flow Chart of Input Interrupt Handling Subroutine

ESI OUTPUT:

The chain of events for output data transfers which operate independently of the program is:

The computer sends an External Function instructing the CTM that output data will be forthcoming. The CTM then presents a Service Request signal to the CTMC. The CTMC then "locks on" to this CTM and sends an Output Data Request to the computer along with the ESI Identifier Word. The computer, upon the receipt of the ODR and ESI Identifier, gets ready to place the data in the ESI buffer on the output data lines. The computer then does an output from the ESI Buffer and sends an Output Acknowledge to the CTM. As a result, the CTM drops its Service Request. This process is repeated until the output CTM receives an end-of-transmission bit. This procedure holds true for all existing output devices.

	EOT	S	ESI DATA WORD 8 (Bits)
35 10	9	8	7 0

EOT (1 bit) = End-of-transmission (Stop the output transfers).

S (1 bit) = Start Bit (used only with low-speed and medium speed CTM's).

ESI Data Word (8 bits).

When output is initiated with CLT's, the above described sequence holds replacing "CTM" by "CLT".

ESI INITIAL LOAD OUTPUT:

05-XX (SZ) Store Zero (zero all ESI addresses).

10-XX (LA) Load A (Load ESI Access Control Word is Register A).

01-XX (SA) Store A (Store the ESI Access Control Word in the ESI address).

75-05 (LOCM) Load Output Channel and Monitor (causes the computer to honor ODR's; set "U" less than 200_8 to speed up the execution time).

75-10 (LFC) Load Function in Channel (One-word buffer to tell Communication Subsystem that data is about to be sent). This step is necessary for CLT's only.

75-12 (JFC) Jump Function Channel (Wait until function leaves computer) The time difference between when a function leaves the computer and the time when the next Function Word or output word can be sent to any output device on the channel just activated must be 3.75 microseconds.

This short subroutine is required to establish the ESI buffers. Once they have been established the I/O section operates independently of the program, and therefore, will not require any more program attention until one of the buffers has been filled. At that time a Monitor Interrupt will be generated which the computer will recognize and will handle according to the subroutine in Figure 6-7.

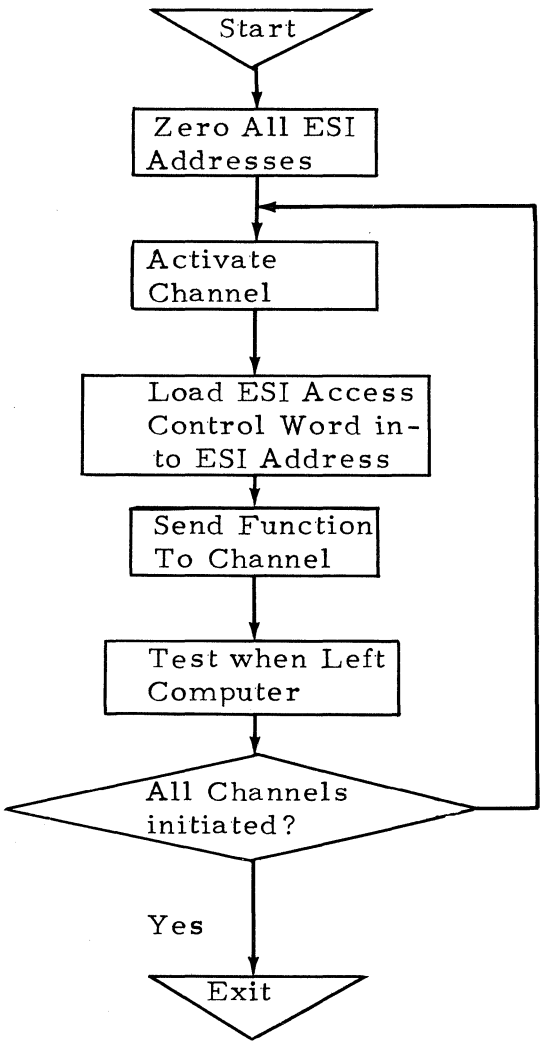


Figure 6-7 Flow Chart of ESI Initial Load Output Subroutine

PROGRAMMING ESI OUTPUT:

When the ESI Monitor Interrupt sends the program to its assigned Interrupt address, the subroutine shown in Figure 6-8 is initiated.

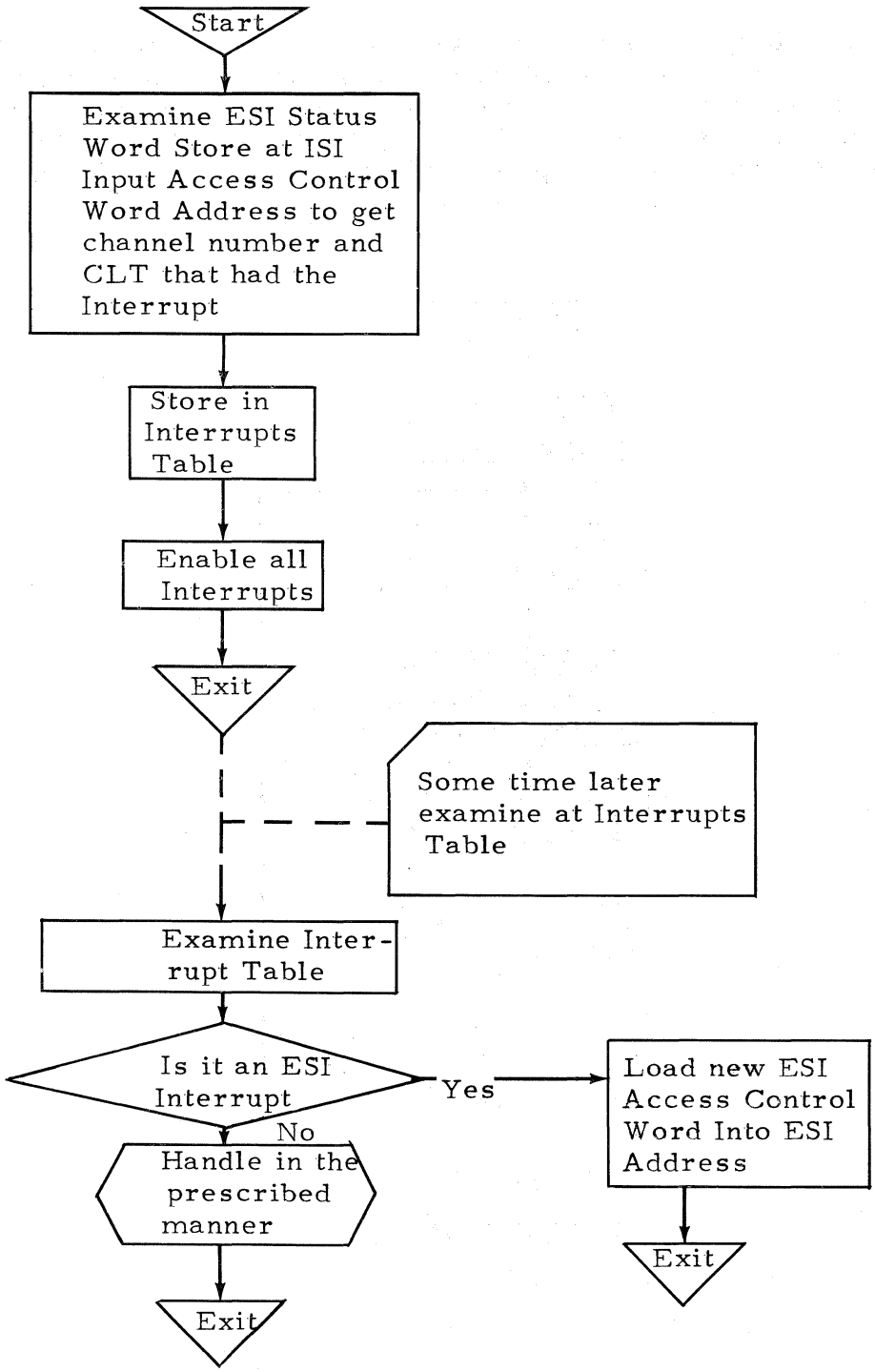


Figure 6-8 Flow Chart of Output Interrupt Handling Subroutine

7. INSTRUCTION REPERTOIRE

FIXED POINT ARITHMETIC

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction	Mnemonic
Add to A	AA

14	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U is added to the contents of A, and the sum is stored in A.

INDICATORS - Carry and Overflow

Example:

Assume $(A)_i$ = 000003564115
(U) = 000001412310

Then $(A)_f$ = 000005176425

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction	Mnemonic
Add Negative to A	ANA

15	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U is subtracted from the contents of A, and the difference is stored in A.

INDICATORS - Carry and Overflow

Example:

Assume $(A)_i$ = 000003 564115
(U) = 000001 412310

Then $(A)_f$ = 000002 151605

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Add Magnitude to A AM, AMA

16	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Add the absolute value of the contents of U to the algebraic value of the contents of A, and store the sum in A.

INDICATORS - Carry and Overflow

Example:

Assume $(A)_i$ = 000032 123456
(U) = 002244 663311

Then $(A)_f$ = 002277 006767

If (U) = 775533 114466, $(A)_f$ would be the same as above.

Execution Time in usec.

Control Registers	Alternate Bank	Same Bank
.75	.75	1.50

Instruction	Mnemonic
Add Negative Magnitude to A	ANM, ANMA

17	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Subtract the absolute value of the contents of U from the algebraic value of the contents of A, and store the difference in A.

INDICATORS - Carry and Overflow

Example:

Assume $(A)_i$ = 000002 572043
(U) = 000000 405226

Then $(A)_f$ = 000002 164615

If (U) = 777777 372551, $(A)_f$ would be the same as above.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction	Mnemonic
Add Upper	AU

20	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U are algebraically added to the contents of A. The sum is stored in A+1. The original contents of U and the contents of A are unaffected.

INDICATORS - Carry and Overflow

Example:

Assume $(A)_i$ = 000026 653211
(U) = 000000 327654

Then $(A+1)_f$ = 000027 203065

Execution Time in usec.

Control Registers	Alternate Bank	Same Bank
.75	.75	1.50

Instruction	Mnemonic
Add Negative Upper	ANU

21	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U are algebraically subtracted from the contents of A. The difference is stored in A+1. The original contents of U and the contents of A are unaffected.

INDICATORS - Carry and Overflow

Example:

Assume $(A)_i$ = 000003 564115
(U) = 000001 412310

Then $(A+1)_f$ = 000002 151605

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Add to X AX

24	j	X	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Add the contents of the X-register designated by the a-field and the contents of U. Store the sum in the specified X_a Register.

INDICATORS - Carry and Overflow

Example:

Assume $(X_a)_i$ = 000001 034444
(U) = 000001 000200

Then $(X_a)_f$ = 000002 034644

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction	Mnemonic
Add Negative to X	ANX

25	j	X	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U are subtracted from the contents of the X Register designated by the a-field; the difference is stored in X_a.

INDICATORS - Carry and Overflow

Example:

Assume $(X_a)_i$ = 000001 005236
(U) = 000000 000625

Then $(X_a)_f$ = 000001 004411

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.375	3.125

Instruction	Mnemonic
Multiply Integer	MI

30	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A are algebraically multiplied by the contents of U. The most significant 36 bits of the product are stored in the designated A-register. The least significant 36 bits of the product are stored in A+1.

Example:

Assume $(A)_i$ = 000000 520331
(U) = 000000 000032

Then $(A,A+1)_f$ = 000000 000000 000021 053012

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.375	3.125

Instruction	Mnemonic
Multiply Single Integer	MSI

31	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A are algebraically multiplied by the contents of U. The least significant 36 bits of the product are stored in the designated A-register. The most significant 36 bits of the product are lost.

Example:

Assume $(A)_i$ = 000000 520331
(U) = 000000 000032

Then $(A)_f$ = 000021 053012

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.375	3.125

Instruction	Mnemonic
Multiply Fractional	MF

32	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A are algebraically multiplied by the contents of U. The most significant 36 bits of the product are stored in the designated A-register. The least significant 36 bits of the product are stored in A+1. The 72 bit product is arithmetically shifted left 1 place before storing in A and A+1.

Assume $(A)_i$ = 312000 000000
(U) = 000000 004444

Then $(A, A+1)_f$ = 000000 003466 320000 000000

	Execution Time in usec.	
	Alternate Bank	Same Bank
	10.125	10.875

Instruction	Mnemonic
Divide Integer	DI

34	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A, A+1 are algebraically divided by the contents of U. The quotient is stored in the designated A-register. The remainder, with the sign of the dividend, is stored in A+1.

Example:

Assume $(A,A+1)_i$ = 000000 000000 000061 026335
(U) = 000000 001300

Then $(A,A+1)_f$ = 000000 043526 000000 000135

	Execution Time in usec.	
	Alternate Bank	Same Bank
	10.125	10.875

Instruction	Mnemonic
Divide Single Fractional	DSF

35	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A, A+1 are algebraically divided by the contents of U. The quotient is arithmetically shifted right one place and stored in A+1. The remainder is lost.

Example:

Assume $(A, A+1)_i$ = 000000 000014 000000 000000
(U) = 000000 000132

Then $(A+1)_f$ = 042104 210421

	Execution Time in usec.	
	Alternate Bank	Same Bank
	10.125	10.875

Instruction	Mnemonic
Divide Fractional	DF

36	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A, A+1 are algebraically divided by the contents of U. The quotient is arithmetically shifted right one place and stored in the designated A register. The remainder, with the original sign of the dividend, is stored in A+1.

Example:

Assume $(A, A+1)_i$ = 000000 000000 000061 026335
(U) = 000000 001300

Then $(A, A+1)_f$ = 000000 021653 000000 000135

	Execution Time in usec.	
	Alternate Bank	Same Bank
	1.625	2.375

Instruction	Mnemonic
Double Precision Fixed Point Add	DA

71	10	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U, U+1 are algebraically added to the (A, A+1). The sum is stored in A, A+1.

INDICATORS - Carry and Overflow

Assume $(A,A+1)_i$ = 123001 230121 400002 321021
$(U, U+1)$ = 000011 112431 456321 000105

Then $(A, A+1)_f$ = 123012 342553 056323 321126

	Execution Time in usec.	
	Alternate Bank	Same Bank
	1.625	2.375

Instruction	Mnemonic
Double Precision Fixed Point Add Negative	DAN

71	11	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U, U+1 are algebraically subtracted from the contents of A, A+1. The difference is stored in A, A+1.

INDICATORS - Carry and Overflow

Example:

Assume $(A,A+1)_i$ = 000000 543210 210056 523004
(U, U+1) = 000000 430100 000042 110002

Then $(A,A+1)_f$ = 000000 113110 210014 413002

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction	Mnemonic
Add Halves	AH

72	04	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The least significant halves of the contents of U and the contents of A are algebraically added; the most significant halves of the contents of U and the contents of A are added. The sums are stored in the corresponding halves of the designated A register.

Example:

Assume $(A)_i$ = 000123 555123
(U) = 000001 223000

Then $(A)_f$ = 000124 000124

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction	Mnemonic
Add Negative Halves	ANH

72	05	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The value in each half of the contents of U is subtracted from the value in the corresponding halves of the specified A Register. There is no interaction between the upper and lower halves. The differences are stored in the A-Register.

Example:

Assume $(A)_i$ = 000123 555123
(U) = 000001 223000

Then $(A)_f$ = 000122 332123

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction	Mnemonic
Add Thirds	AT

72	06	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Each third of the contents of U is algebraically added to each corresponding third of the contents of A. The sums are stored in the designated A register.

Example:

Assume $(A)_i$ = 0001 2355 5123
(U) = 0000 0122 3000

Then $(A)_f$ = 0001 2477 0124

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction	Mnemonic
Add Negative Thirds	ANT

72	07	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Each third of the contents of U are algebraically subtracted from the corresponding thirds of the contents of A. The differences are stored in the designated A register.

Example:

Assume $(A)_i$ = 0001 2355 5123
(U) = 0000 0122 3000

Then $(A)_f$ = 0001 2233 2123

FLOATING POINT ARITHMETIC

The purpose of floating point operations is to allow more efficient computation of numerical data with a wide range of magnitude. This is made possible by expressing the data in floating point number format.

A floating point number represents the value of a real number in the form of a signed binary fraction and a biased exponent (characteristic). The exponent is the power to which the base (2) must be raised so that when multiplied by the fraction, the product represents the value of the real number being expressed. A bias number added to the exponent forms the characteristic which indicates either a positive or a negative exponent.

A floating point number is normalized if the high order bit position of the fraction contains a significant, or non-sign, bit. To change the algebraic sign of a floating point number, the entire floating point number is complemented.

Floating point numbers are expressed in single or double precision format. Special instructions provide for certain format manipulations such as changing floating point operand precision, packing and unpacking of floating point numbers, and the determination of the difference between two characteristics.

In the 1108 floating point arithmetic, the mantissa (numerical value of the data) is always considered to be a fraction. When normalized, the leading bit of the mantissa is equal to 1 for a positive mantissa. A negative mantissa is normalized when the leading bit of the mantissa is equal to 0. Therefore, the value of a normalized mantissa will be equal to or greater than 1/2, but less than 1.

Floating-Point Word Formats - The UNIVAC 1108 processor can operate with two forms of floating-point arithmetic: single-precision and double-precision.

Single-precision instructions produce double-precision results, i.e., a two-word result. Double-precision instructions also produce double-length results (72-bits).

The 1108 requires that the mantissas and the characteristics with their separate signs be provided in the following formats.

Single-Precision Floating-Point Format -

SIGN	CHARACTERISTIC (8 bits)	MANTISSA (27 bits)
35	34 27	26 0

The mantissa is the numerical value of the data, it is always a fractional value less than 1. However, the characteristic is not the exponent of the mantissa; it is the exponent of the base. For example, the number 8 can be represented as a fraction times the power of 10; thus, $.08 \times 10^2$. In 1108 floating-point notation, 8 is expressed as a fraction times a power of 2. Therefore, $8 = .5 \times 2^4$. In this case, 4 is the characteristic, the exponent of the base 2.

Double-Precision Floating-Point Format -

SIGN	CHARACTERISTIC (11 bits)	MANTISSA (60 bits)
71	70 60	59 36 35 0

The same information that pertains to the single-precision floating-point format also pertains to the double-precision floating-point format.

Sign Representation For Floating-Point Arithmetic - Both the characteristic and mantissa for floating-point arithmetic (single or double) may represent positive or negative values.

Positive values of the mantissa are represented by a zero in the sign position and the exact numerical value of the data. For the mantissa to be negative, the procedure is as follows:

(1) Represent the mantissa as a positive value with no regard to the negative sign.

(2) Represent the characteristic according to the rules of Characteristic Biasing as explained in the following pages.

(3) One's complement the entire floating-point word (single-precision, 36 bits, double-precision, 72 bits).

To avoid using two separate signs (one for the characteristic and one for the mantissa) within the same word, characteristic biasing is employed to indicate the sign of the characteristic.

Single Precision:

For single-precision, characteristic biasing consists of adding to the true or unbiased characteristic the bias value of $128(200_8)$. The 8-bit characteristic permits a range of -128 (-200_8 to $+177_8$) as shown in the following table.

Characteristic Values

Decimal		Octal	
True	Biased	True	Biased
-128	000	-200	000
000	128	000	200
+127	255	177	377

In programming, true values are coded; the assembler provides the bias. If after assembly the program changes a number from fixed-point to floating-point, the program must also provide the proper bias.

To illustrate the principles involved, the value $.75_{10} \times 2^3$ is presented with every possible combination of signs.

(1) $.75 \times 2^3$	$.75 \times 2^3$ (unbiased)	= 003	600	000	000_8
	Bias	= 200			
	$.75 \times 2^3$	= 203	600	000	000
(2) $-.75 \times 2^3$	$.75 \times 2^3$ (unbiased)	= 003	600	000	000
	Bias	= 200			
	$.75 \times 2^3$	203	600	000	000
	1's Complement				
	$-.75 \times 2^3$	574	177	777	777
(3) $.75 \times 2^{-3}$	$.75 \times 2^{-3}$ (unbiased)	= (-3)	600	000	000
	Bias	200			
	$.75 \times 2^{-3}$	175	600	000	000

(4) $-.75 \times 2^{-3}$	$.75 \times 2^{-3}$ (unbiased) =	(-3)	600	000	000
	Bias	200			
		175	600	000	000
	1's Complement				
	$-.75 \times 2^{3}$	602	177	777	777

Double Precision:

For double-precision, the true or unbiased characteristic is added to a bias value of 1024 (2000). The 11-bit characteristic permits range of values show below:

Characteristic Values

Decimal		Octal	
True	Biased	True	Biased
-1024	0000	-2000	0000
0000	1024	0000	2000
+1023	2047	1777	3777

In programming, true values are coded; the assembler provides the bias. If after assembly the program changes a number from fixed-point to floating-point, the program must also provide the proper bias.

(1) $.75 \times 2^{3}$	$.75 \times 2^{3}$ (unbiased)	= 0003	60 000 000 000 000 000 000$_8$
	Bias	= 2000	
	$.75 \times 2^{3}$	= 2003	60 000 000 000 000 000 000
(2) $-.75 \times 2^{3}$	$.75 \times 2^{3}$ (unbiased)	= 0003	60 000 000 000 000 000 000
	Bias	= 2000	
	$.75 \times 2^{3}$	= 2003	60 000 000 000 000 000 000
	1's Complement		
	$-.75 \times 2^{3}$	= 5774	17 777 777 777 777 777 777
(3) $.75 \times 2^{-3}$	$.75 \times 2^{-3}$ (unbiased)	= (-3)	60 000 000 000 000 000 000
	Bias	= 2000	
	$.75 \times 2^{-3}$	= 1775	60 000 000 000 000 000 000

(4) $-.75 \times 2^{-3}$	$.75 \times 2^{3}$ (unbiased)	= (-3)	60 000 000 000 000 000 000
	Bias	= 2000	
	$.75 \times 2^{-3}$	= 1775	60 000 000 000 000 000 000
	1's Complement		
	$-.75 \times 2^{-3}$	= 6002	17 777 777 777 777 777 777

Instruction	Mnemonic	Execution Time in usec. I/D In Alt. Banks	I/D In Same Bank
Floating Add	FA	1.750	2.500

76	00	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 0

Description

The floating point number stored in U is added to the floating point number in A. The normalized sum replaces the addend in A. The unnormalized residue is stored in the next higher A register; see below for definition of residue.

Execution

1. The absolute values of the characteristics, $(A)_{34-27}$ and $(U)_{34-27}$, are compared. If they are unequal, the fraction with the smaller characteristic, either $(A)_{26-0}$ or $(U)_{26-0}$, is transmitted and right justified in a non-addressable register, S; the sign bit is also sent. If the characteristics are equal, $(U)_{26-0}$ is sent to S.

The most significant part of the register, $(S)_{35-27}$, is sign bit extended to conform to the algebraic sign of the fraction contained in S. Associated with S is the non-addressable register (Q). Q is filled with the same sign bits as the sign of the fraction in S.

2. An arithmetic right shift is executed in S. Bits shifted out of S are shifted into the most significant end of Q. The shift is terminated when the shift count equals the difference in the absolute value of the characteristics.

3. The fraction with the larger characteristic is then algebraically added to $(S)_{27-0}$. Q is not involved in the addition; it contains the residue.

4. If the fraction sum produces a high order carry to bit position 28 in S, the contents of S and Q are algebraically right shifted one place and the absolute value of the larger characteristic is increased by 1.

5. The residue bits in Q are then repacked to floating point format by 1) shifting Q 9 places to the right, 2) subtracting (33_8) from the absolute value of the characteristic, 3) since the sign associated with the residue is the sign of the fraction sent to S, this sign is inserted as the sign bit in (Q) 35, 4) then inserting the resultant characteristic, or its complement if the sign bit is negative, in $(Q)_{34-27}$. The residue now in floating point format, is transmitted to the next higher arithmetic register, A+1.

6. If after the fraction addition the sum, $(S)_{35-0}$, is not normalized, the sum is arithmetically left shifted until either $(S)_{27} \neq (S)_{26}$ or a shift count of (33_8) is achieved (indicating a zero fraction sum). The absolute value of the characteristic is reduced by the amount of the shift.

7. The fraction sum, $(S)_{26-0}$, and the sign bit are packed with the larger characteristic and sent to the A register in sign form.

Example:

Assume $(A)_i$ = 264 423456722
(U) = 250 663543211

Then $(A)_f$ = 264 423545276
$(A + 1)_f$ = 231 321100000

Programming Note:

Interrupts may occur due to characteristic underflow or overflow.

Execution Time in usec.	
Alternate Bank	Same Bank
1.750	2.500

Instruction	Mnemonic
Floating Add Negative	FAN

76	01	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The floating point number stored in U is subtracted from the floating point number in A. The normalized difference replaces the minuend in A. An unnormalized residue is stored in the next higher A register.

Execution:

The same as FA except the U operand is complemented before Step 1 is executed.

Example:

Assume $(A)_i$ = 275 660000011
(U) = 250 420000002
Then $(A)_f$ = 275 657777747
$(A+1)_f$ = 535 777777577

Programming Note:

Interrupts may occur due to characteristic underflow or overflow.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.625	3.375

Instruction	Mnemonic
Double Precision Floating Add	DFA

76	10	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description

The double precision floating point number in U, U + 1 is added to the double precision floating point number in A, A + 1. The normalized, double precision, sum replaces the addend in A, A + 1.

Execution

1. The absolute values of the characteristics, $(A, A+1)_{70-60}$ and $(U, U+1)_{70-60}$, are compared; if they are unequal the fraction with the smaller characteristic, either $(A, A+1)_{59-0}$ or $(U, U+1)_{59-0}$, is transmitted and right justified in a non-addressable double-length register, D. If the characteristics are equal, $(U, U+1)_{59-0}$ is sent to D. The most significant part of the register, $(D)_{71-60}$, is sign bit extended to conform to the sign form of the fraction in D.

2. An arithmetic right shift is executed in D. Bits shifted out of the least significant bit position are lost. The shift is terminated when the shift count equals the difference in the absolute values of the characteristics.

3. The fraction with the larger characteristic is algebraically added to $(D)_{59-0}$.

4. If the fraction sum produces a high order carry to bit position 60 of D, the contents of D are arithmetically right shifted 1 place and the absolute value of the larger characteristic is increased by 1.

5. If, after the fraction addition, the sum is not normalized, the sum is arithmetically left shifted until either $(D)_{60} \neq (D)_{59}$ to produce a normalized result, or until a shift count of (74_8) is achieved (indicating a zero fraction). The absolute value of the larger characteristic is reduced by the amount of the shift.

6. When the fraction sum is normalized in D, the larger characteristic is transmitted to $(D)_{70-60}$. The resulting double precision floating point number is transmitted, in sign form, to A, A + 1; $(D)_{71-36}$ to A and $(D)_{35-0}$ to A + 1.

Example:

Assume	$(A),(A+1)_i$	=	2002 32164567767234567012
Assume	$(U),(U+1)$	=	6000 43245761102345321134
Then	$(A),A+1)_f$	=	2001 55222754177162642254

Programming Notes:

1. Interrupts may occur due to characteristic underflow or overflow.

2. A fraction sum yielding zero value causes A, A + 1 to contain all zero bits.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.625	3.375

Instruction	Mnemonic
Double Precision Floating Add Negative	DFAN

76	11	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The double precision floating point number stored in U, U + 1 is subtracted from the double precision floating point number in A, A + 1. The normalized double precision difference replaces the minuend in A, A + 1.

Execution:

Same as DFA except the U, U + 1 operand is complemented before execution.

Programming Notes:

1. Interrupts may occur due to characteristic underflow or overflow.

2. A fraction difference yielding zero value causes A, A + 1 to contain all zero bits.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.625	3.375

Instruction	Mnemonic
Floating Multiply	FM

76	02	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The floating point number stored in U is multiplied by the floating point number in A. The most significant part of the product is stored in A; the least significant part in A + 1. Both parts are in floating point format; the least significant characteristic is (33_8) less than the most significant characteristic.

Execution:

1. The A register contains the multiplier; the contents of U is the multiplicand. The 27 bit fractions, $(A)_{26-0}$ and $(U)_{26-0}$ are sent, right justified, to non-addressable registers S and Q; the sign bits are also sent. The fractions in S and Q are sign bit extended to correspond to the sign bit of the transmitted fractions.

2. If a multiplier or multiplicand is negative, it is complemented. The anticipated algebraic sign of the product is saved and is used later to automatically determine the sign of the fraction product. The fraction product is then formed according to the rules for fixed point multiplication in what may

be considered a double length register consisting of S and Q; S is the most significant register.

3. If $(S)_{26} \neq (S)_{27}$ the product is normalized and 54 bits long; if $(S)_{26} = (S)_{27}$, the product is 53 bits long and it is arithmetically shifted one place to the left. If the multiplier and the multiplicand were normalized, the product is normalized.

4. The characteristics $(A)_{34-27}$ and $(U)_{34-27}$ are added. Then either (200_8) is subtracted from the characteristic sum or, if the product is 53 bits long, (201_8) is subtracted.

5. The most significant bits of the product, $(S)_{27-0}$, and the sign bit are packed with the resultant characteristic in the designated A register to form the most significant product word. The A register contains the product word in sign form.

6. The least significant product bits, $(Q)_{35-9}$, are shifted 9 places. Then (33_8) is subtracted from the characteristic computed in Step 4; the results are packed in $(Q)_{34-27}$. The contents of Q are transferred in sign form to the A + 1 to form the least significant product word.

Programming Notes:

1. If the multiplier or multiplicand is unnormalized, the product is not normalized.

2. Interrupts may occur due to characteristic underflow or overflow.

Example:

Assume $(A)_i$ = 211 654321001
(U) = 172 650454045

Then $(A)_f$ = 203 543210122
$(A+1)_f$ = 150 177541045

	Execution Time in usec.	
	Alternate Bank	Same Bank
	4.250	5.000

Instruction	Mnemonic
Double Precision Floating Multiply	DFM

76	12	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The double precision floating point number stored in U, U+1 is multiplied by the double precision floating point number in A, A+1. The double precision product is stored in A, A+1.

Execution:

1. The designated A register contains the most significant part of the double precision multiplier operand; the A+1 register contains the least significant part. The multiplicand is in U and U+1. The 60 bit fractions, $(A, A+1)_{59-0}$ and $(U, U+1)_{59-0}$, are right justified and sent, including sign bits, to non-addressable double length registers; each fraction is sign bit extended in the registers to conform with the sign of the transmitted fraction.

2. The anticipated algebraic sign of the product is stored and is later used to automatically determine the sign of the product fraction. The resulting product is formed and right justified in a double length register, D.

3. If $(D)_{59} \neq (D)_{60}$, the product is normalized and 60 bits long; if $(D)_{59} = (D)_{60}$, the product is 59 bits long and is arithmetically shifted one place to the left. If the multiplier and multiplicand are normalized, the product is normalized.

4. The characteristics, $(A, A+1)_{70-60}$ and $(U, U+1)_{70-60}$, are added. Then either (2000_8) is subtracted from the characteristic sum, or, if the product is 59 bits long, (2001_8) is subtracted.

5. The fraction product, $(D)_{59-36}$, and the sign bit are packed with the resultant characteristic in the designated A register to form the most significant part of the double precision product. $(D)_{35-0}$ is transmitted to the A + 1 register to form the least significant part of the double precision product.

Example:

Assume	$(A), (A+1)_i$	=	2032 44444444444444444444
	$(U), (U+1)$	=	1754 66666666666666666666
Then	$(A), (A+1)_f$	=	2005 76543207654320765430

Programmer's Note:

1. If the multiplier or multiplicand is unnormalized the product may not be normalized.

Instruction	Mnemonic	Execution Time* in usec. Control Registers	Alternate Bank	Same Bank
Floating Divide	FD		8.250/8.50	9.000/9.25

*The longer time is needed if the absolute value of the mantissa of (A) is greater than the absolute value of the mantissa of (U).

76	03	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The normalized floating point number stored in the designated A register is divided by the normalized floating point number in U. The normalized quotient replaces the dividend in the A register and the unnormalized remainder is right justified and stored in the A + 1 register. The sign of the remainder is the sign of the dividend.

Example:

Assume $(A)_i$ = 203 543210123
(U) = 211 654321011

Then $(A)_f$ = 172 650454035
$(A+1)_f$ = 150 636706373

Programming Notes:

1. If the divisor in U has a zero fraction, execution of the instruction results in a divide fault and an interrupt to location 247 occurs. The contents of A and U are unchanged.

2. Unnormalized divisors or dividends should not be used since they may yield incorrect results.

3. Interrupts may occur due to characteristic underflow or overflow.

Instruction	Mnemonic	Execution Time* in usec. Control Registers	Alternate Bank	Same Bank
Double Precision Floating Divide	DFD		17.250/17.50	18.000/18.25

*The longer time is needed if the absolute value of the mantissa portion of (A, A+1) is greater than the absolute value of the mantissa portion of (U, U+1).

76	13	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The normalized double precision floating point number in A, A + 1, is divided by the normalized double precision floating point number in U and U + 1. The normalized quotient replaces the dividend in A, A + 1.

Example:

Assume	$(A), (A+1)_i$	=	2004 42220 00000 00000 00000
	(U), (U+1)	=	2000 53300 00000 00000 00000
Then	$(A), (A+1)_f$	=	2004 62452 032436176711620

Programming Notes:

1. If the divisor in U, U + 1 has a zero fraction, in either sign form, execution of the instruction results in a divide fault and an interrupt to location 247 occurs.

2. Unnormalized divisors or dividends should not be used since they may yield incorrect results.

3. Interrupts may occur due to characteristic underflow or overflow.

4. A fraction quotient yielding a zero value causes A, A + 1 to contain all zero bits.

		Execution Time in usec.		
		Control Registers	Alternate Bank	Same Bank
		.750	.750	1.500
Instruction	Mnemonic			
Load and Unpack Floating	LUF			

76	04	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The floating point number, (U), is unpacked, the absolute value of the characteristic, $(U)_{34-27}$, is transmitted and right justified in the designated A register; $(A)_{35-8}$ is zero filled. The fraction, $(U)_{26-0}$, is stored in sign form and right justified in the A + 1 register. The sign of the unpacked fraction is the same as the algebraic sign of the (U).

Example:

Assume (U) = 264 423456722

Then $(A)_f$ = 000 000000264
$(A+1)_f$ = 000 423456722

Programming Note:

No characteristic underflow or overflow can occur.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	1.500	2.250

Instruction	Mnemonic
Double Load and Unpack Floating	DFU

76	14	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description

The floating point number, (U, U+1), is unpacked. The absolute value of the characteristic, $(U, U+1)_{70-60}$, is transmitted and right justified in the designated A register; $(A)_{35-11}$ is zero filled. The high order part of the fraction $(U, U+1)_{59-36}$ is stored in sign form, and right justified, in the A + 1

register; the rest of the fraction $(U, U+1)_{35-0}$ is stored in the A + 2 register. The sign of the unpacked fraction in A + 1, A + 2 is the same as the algebraic sign of (U, U+1).

Example:

Assume	(U, U+1)	=	216363456742 641147534415
Then	$(A)_f$	=	000000002163
	$(A+1)_f$	=	000063456742
	$(A+2)_f$	=	641147534415

Programming Note:

1. No characteristic underflow or overflow can occur.

Execution Time in usec.	
Alternate Bank	Same Bank
1.125	1.875

Instruction	Mnemonic
Load and Convert to floating	LCF

76	05	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description

The absolute value of the characteristic $(A)_{7-0}$, or its complement if (U) is negative, and the sign of the fraction stored in U are packed as a normalized floating point number in A+1. The packed number carries the sign of the fraction in U. The fraction in U is not required to be normalized; see Execution and Programmer's Note.

Execution

1. The (U), which contains the fraction is transmitted to a non-addressable register, Q.

2. If the fraction is not normalized in Q an attempt to normalize is made by shifting (Q) arithmetically up to 27 places left or up to 8 places right. Bits shifted out of Q are lost.

3. When the fraction is either normalized or shifted 27 places to the left, the associated absolute value of the characteristic is either incremented or decremented by the number of shifts respectively made to the right or left.

4. The fraction and the characteristic are packed, and stored in A+1.

Example:

Assume (A) = 000 000000256
(U) = 000 056217704

Then $(A+1)_f$ = 253 562177040

Programming Note:

If an underflow occurs with a non-zero fraction in U, the A + 1 register is unchanged; an interrupt to location 245 results. If an underflow occurs with a zero fraction in U, the A + 1 register is loaded with zero.

If an overflow occurs, the A + 1 register is unchanged; an interrupt to location 246 results.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.125	2.875

Instruction	Mnemonic
Double Load and Convert to Floating	DLCF

76	15	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The absolute value of the characteristic, $(A)_{10-0}$, or its complement if (U, U+1) is negative, and the sign of the double precision fraction, (U, U+1), are packed into a double precision, normalized, floating point number in A + 1, A + 2. The packed number carries the sign of the double precision fraction, (U, U+1). The fraction in U, U + 1 is not required to be normalized; see Execution and Programmer's Note.

Execution:

1. The (U, U+1) is transmitted to a non-addressable double length register, D.

2. If the fraction is not normalized in D an attempt to normalize is made by arithmetically shifting (D) up to 60 places left or up to 11 places right. Bits shifted out of D are lost.

3. When the fraction is either normalized or shifted 60 places to the left, the associated absolute value of the characteristic is either incremented or decremented the number of shifts respectively made to the right or left.

4. The fraction $(D)_{59-0}$ and the characteristic are packed and put in (A+1, A+2).

Example:

Assume (A) = 000000002101
(U, U+1) = 000543212345671122334455

Then $(A,A+1)_f$ = 210454321234567112233445

Programming Notes:

1. If characteristic underflow occurs with a non zero fraction in U and U + 1, the A + 1 and A + 2 registers are unchanged and an interrupt to location 245 results. If a characteristic underflow occurs with a zero fraction in U and U + 1, all zero bits are stored in A + 1, A + 2; no interrupt occurs.

2. If characteristic overflow occurs the A + 1 and A + 2 registers are unchanged and an interrupt to location 246 results.

3. If the fraction in U and U + 1 has a zero value, all zero bits are stored in A + 1, A + 2.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	1.000	1.750

Instruction	Mnemonic
Floating Expand and Load	FEL

76	16	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

This instruction expands a single precision floating point number, the (U), into a double precision floating point number in A, A + 1. The sign of (A, A + 1) is the sign of the (U). The double precision result is normalized only if the single precision number is normalized.

Example:

Assume (U) = 264 423456722

Then $(A,A+1)_f$ = 2064 42345672 200000 000000

Programming Notes:

1. No characteristic underflow or overflow can occur.

2. A single precision floating point number with a zero fraction is converted to all zero bits in A, A + 1; the (A, A + 1) is algebraically positive.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	1.625	2.375

Instruction	Mnemonic
Floating Compress and Load	FCL

76	17	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

This instruction compresses a double precision floating point number in U, U + 1, into a single precision floating point number, stored in A. The single precision result is normalized only if the double precision floating point number is normalized. Characteristic overflow occurs if the unbiased exponent is greater than 127; characteristic underflow if the unbiased exponent is less than 128.

Example:

Assume (U,U+1) = 2051 63456711456712345677

Then $(A)_f$ = 251 634567114

Programming Notes:

1. Since only the 27 high order fraction bits from the (U, U+1) are utilized in A, the floating point number in U and U+1 should be normalized to preserve maximum significance.

2. If characteristic underflow occurs with a non-zero fraction in (U, U+1), the A register is unchanged; an interrupt to location 245 results. If characteristic underflow occurs with a zero fraction, zero bits are stored in the A register; no interrupt occurs.

3. If characteristic overflow occurs, the A register is unchanged and an interrupt to location 246 results.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.750	.750	1.500

Instruction Mnemonic

Magnitude of Characteristic Difference to Upper — MCDU

76	06	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The absolute value of the characteristic, $(U)_{34-27}$, is subtracted from the absolute value of the characteristic, $(A)_{34-27}$. The absolute value of the difference is stored, right justified, in A + 1.

Example:

Assume (A) = 610 327365000
(U) = 205 532475022

Then $(A+1)_f$ = 000 000000016

Programming Notes:

1. Double precision operands yield incorrect results.

2. No characteristic underflow or overflow can occur.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.750	.750	1.500

Instruction Mnemonic

Characteristic Difference to Upper — CDU

76	07	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The absolute value of the characteristic, $(U)_{34-27}$, is subtracted from the absolute value of the characteristic, $(A)_{34-27}$; the difference is right justified in A + 1.

Programming Notes:

1. Double precision operands yield incorrect results.

2. No characteristic underflow or overflow can occur.

Example:

Assume (A) = 257 532475022
(U) = 245 427365000

Then $(A+1)_f$ = 000000 000012

Data-Transfer

The Data Transfer instructions make possible the transfer of information from a set of program referenced storage locations or control registers to another set of referenced storage locations or control registers. Since the transfer is always from or to the arithmetic section of the processor, the transfer always references at least one control register.

The 13 transfer operations are divided into two groups; load instructions and store instructions.

There are four operations which enable the programmer to refer to a storage address plus the next sequential address. These instructions are termed double loads and double stores. The contents of the effective operand address and that address plus one are transferred to or from the designated A and A+1 registers. The j-designator assumes the role of minor function code.

The nine remaining instructions interpret the j-designator as a whole or partial word transfer or defines the u-field of the instruction as the operand.

		Execution Time in usec.		
		Control Registers	Alternate Bank	Same Bank
		.75	.75	1.50
Instruction	Mnemonic			
Load A	LA			

10	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U is transferred to the designated A register.

Example:

Assume (U) = 013246 753135

Then $(A)_f$ = 013246 753135

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Load Negative A LN, LNA

11	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The complement of the contents of U is transferred to the designated A register.

Example:

Assume (U) = 013246 753135

Then $(A)_f$ = 764531 024642

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Load Magnitude A LM, LMA

12	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The absolute value of the contents of U is transferred to the designated A register.

Example:

Assume (U) = 457626 353660

Then $(A)_f$ = 320151 424117

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction	Mnemonic
Load Negative Magnitude A	LNMA

13	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U are transferred to the designated A register. The value in A is always negative.

Example:

Assume (U) = 357626 353660

Then $(A)_f$ = 420151 424117

Execution Time in usec.	
Alternate Bank	Same Bank
1.50	2.25

Instruction Mnemonic

Double Store A DS

71	12	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A, A+1 are transferred to U, U+1.

Example:

Assume (A,A+1) = 123456 002460 431021 065432

Then $(U,U+1)_f$ = 123456 002460 431021 065432

Execution Time in usec.	
Alternate Bank	Same Bank
1.50	2.25

Instruction	Mnemonic
Double Load A	DL

71	13	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U, U+1 are transferred to the designated A, A+1.

Example:

Assume (U,U+1) = 432010 123456 000123 456700

Then $(A,A+1)_f$ = 432010 123456 000123 456700

	Execution Time in usec.	
	Alternate Bank	Same Bank
	1.50	2.25

Instruction	Mnemonic
Double Load Negative A	DLN

71	14	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The complement of the contents of U, U+1 is transferred to the designated A, A+1.

Example:

Assume (U,U+1) = 432010 123456 000123 456700

Then $(A,A+1)_f$ = 345767 654321 777654 321077

	Execution Time in usec.	
	Alternate Bank	Same Bank
	1.50	2.25

Instruction	Mnemonic
Double Load Magnitude A	DLM

71	15	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The absolute value of the contents of U, U+1 is transferred to the designated A, A+1.

Example:

Assume (U,U+1) = 432100 765123 456701 332104

Then $(A,A+1)_f$ = 345677 012654 321076 445673

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Store A SA

01	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A are transferred to U.

Example:

Assume (A) = 001133 445566

Then $(U)_f$ = 001133 445566

Note: j = 00 to 15_8. If j = 16_8 or 17_8, no storage results.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Store Negative A SN, SNA

02	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The complement of the contents of A is transferred to U.

Example:

Assume (A) = 001133 445566

Then $(U)_f$ = 776644 332211

Note: j = 00 to 15_8. If j = 16_8 or 17_8 no storage results.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Store Magnitude A SM, SMA

03	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The absolute value of the contents of A is transferred to U.

Example:

Assume (A) = 654321 012345

Then $(U)_f$ = 123456 765432

Note: j = 00 to 15_8. If j = 16_8 or 17_8 no storage results.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction — Mnemonic

Store Zero — SZ

05	j		x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Zero bits are stored in U; the sign is positive.

Example:

Assume $(U)_i$ = 111222 333444

Then $(U)_f$ = 000000 000000

Programming Note:

The a-designator is ignored.

j = 00 to 15_8. If j = 16_8 or 17_8 no storage results.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Store R SR

04	j	R	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of the R register specified by the a-designator are transferred to U.

Example:

Assume (R_a) = 123123 456456

Then $(U)_f$ = 123123 456456

Note: $j = 00$ to 15_8. If $j = 16_8$ or 17_8 no storage results.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Load R LR

23	j	R	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U are transferred to the designated R_a register.

Example:

Assume (U) = 112233 445566

Then $(R_a)_f$ = 112233 445566

	Execution Time in usec.	
	Alternate Bank	Same Bank
	1.5+1.5K	2.25+1.5K

Instruction	Mnemonic
Block Transfer Repeat	BT

22	j	X	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The X_m portion of the X-register designated in the a-field is used as the address of the word to be moved. Each time the instruction is executed X_i is added to X_m to form a new address. The location to which the data is moved is specified by U. This instruction continues to be executed until the repeat count is reduced to zero.

Shifting

The shift instructions allow for the movement of data within the arithmetic control registers. All shift operations use the j-designator as a minor function code.

The twelve shift instructions are divided into four groups, Circular, Logical, Algebraic, and Scale Factor.

Circular, Logical and Algebraic shifts are program controlled, that is,the shift is executed the number of times specified in the u-designator. The u-designator may be modified by an index register or, with indirect addressing employed, contain the address of the operand. Regardless of the method employed, if the number of bits shifted exceeds 72, invalid results may occur.

On Scale Factor shifts, the value transferred to A is left circular shifted until bit 35 is not equal to bit 34. The number of bits shifted is stored in the next higher A-register, A+1.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Single Shift Circular	SSC

73	00	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of the specified A Register are shifted right U number of binary places. Bits shifted out at the right reappear at the left.

Example:

Assume $(A)_i$ = 000000 012345
u = 000014_8

Then $(A)_f$ = 234500 000001

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Double Shift Circular	DSC

73	01	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of the specified A Register and the next higher register, treated as a single 72 bit word, are shifted right U number of places. Bits shifted out at the right of register A + 1 reappear at the left of register A.

Example:

Assume u = 000024_8

$(A,A+1)_i$ = 000000 000000 123456 765432

Then $(A,A+1)_f$ = 575306 400000 000000 024713

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Single Shift Logical	SSL

73	02	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A are shifted right U number of places. Bits shifted out of the least significant portion of A are lost. Vacated bit positions are zero filled.

Example:

Assume $u = 000002$
$(A)_i = 000123\ 366157$

Then $(A)_f = 000024\ 675433$

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Double Shift Logical	DSL

73	03	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of the specified A Register and the next higher register, treated as a single 72 bit word, are shifted right U number of places. Bits shifted out at the right end of register A + 1 are lost, and zeros replace bits moved out of the most significant positions of register A.

Example:

Assume u = 000011_8

Then $(A,A+1)_i$ = 777666 555444 333222 112345
$(A,A+1)_f$ = 000777 666555 444333 222112

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Single Shift Algebraic	SSA

73	04	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A are shifted right U number of places. Bits shifted from the least significant portion are lost. Vacated bit positions in the most significant portion of A are filled by sign bits.

Example:

Assume $u = 000006$
$(A)_i = 777777\ 771352$

Then $(A)_f = 777777\ 777713$

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Double Shift Algebraic	DSA

73	05	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A, A+1 are shifted right U number of places. Bits shifted from the least significant portion, A+1, are lost. Vacated bit positions in the most significant portion, A, are filled by sign bits.

Example:

Assume u = 000011_8

$(A,A+1)_i$ = 544332 211004 653044 135704

Then $(A,A+1)_f$ = 777544 332211 004653 044135

	Execution Time in usec.	
	Alternate Bank	Same Bank
	1.125	1.875

Instruction	Mnemonic
Load Shift and Count	LSC

73	06	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U are transferred to the designated A register. The value in A is scaled and the shift count is stored in A+1.

Execution:

The (U) are transferred to a shift register and a left circular shift is executed until bit 35 ≠ to bit 34. When this condition is met, the scaled value is transferred to A and the number of bits shifted, the shift count, is stored in A+1.

If the (U) are all zeros or ones, 35 will be stored in A+1.

Example (in binary notation):

Assume (U) = 000000 000000 110010 111011 000101 010001

Then $(A)_f$ = 011001 011101 100010 101000 1000000 000000
$(A+1)_f$ = 000000 000000 000000 000000 0000000 001011

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.125	2.875

Instruction	Mnemonic
Double Load Shift and Count	DLSC

73	07	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U, U+1 are transferred to the designated A and A+1 registers. The value in A, A+1 is scaled and the shift count stored in A+2.

Execution:

The (U,U+1) are transferred to the 2 shift registers. A double left circular shift is performed until bit 71≠bit 70. When this condition is met, the scaled value is transferred to A and A+1. The number of bits shifted, the shift count, is stored in A+2. If the (U, U+1) are all ones or zeros, 71 will be stored in A+2.

Example:

Assume (U,U+1) = 003456 765432 111234 444555

Then $(A,A+1)_f$ = 345676 543211 123444 455500
$(A+2)_f$ = 000000 000006

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Left Single Shift Circular	LSSC

73	10	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A are shifted left U number of places. Bits shifted from the most significant portion reappear in the least significant portion of A.

Example:

Assume $u = 000011_8$

$(A)_i = 222333\ 444555$

Then $(A)_f = 333444\ 555222$

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Left Double Shift Circular	LDSC

73	11	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A, A+1 are shifted left U number of places. Bits shifted from the most significant portion of A reappear in the least significant portion of A+1.

Example:

Assume u = 000017_8

$(A,A+1)_f$ = 123450 654321 000222 333444

Then $(A,A+1)_f$ = 065432 100022 233344 412345

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Left Single Shift Logical	LSSL

73	12	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A are shifted left U number of places. Bits shifted from the most significant portion of A are lost. Vacated bit positions are filled by zeros in the least significant portion of A.

Example:

Assume $u = 000014_8$
$(A)_i = 111222\ 333444$

Then $(A)_f = 223334\ 440000$

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Left Double Shift Logical	LDSL

73	13	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of A, A+1 are shifted left U number of places. Bits shifted from the most significant portion of A are lost. Vacated bit positions are filled by zeros in the least significant portion of A+1.

Example:

Assume u = 000011_8

$(A,A+1)_i$ = 112233 445566 776543 210123

Then $(A,A+1)_f$ = 233445 566776 543210 123000

Index Register Transfer

The index transfer instructions make possible the movement of data between main storage and the index registers.

The four index transfer operations are divided into two groups; load instructions and store instructions.

The appropriate index register is coded in the a-designator portion of the instruction word.

		Execution Time in usec.		
		Control Registers	Alternate Bank	Same Bank
		.875	.875	1.625
Instruction	Mnemonic			
Load X Modifier	LXM			

26	j	X	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U are transferred to the lower half of the designated X_a register. The upper half of the X_a register is unaffected.

Example:

Assume (U) = 333444 222111
$(X_a)_i$ = 123456 234567

Then $(X_a)_f$ = 123456 222111

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Load X LX

27	j	X	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U are transferred to the designated X_a register.

Example:

Assume (U) = 054321 066113

Then $(X_a)_f$ = 054321 066113

	Execution Time in usec.	
	Alternate Bank	Same Bank
	1.0	1.75

Instruction	Mnemonic
Load X Increment	LXI

46	j	X	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of U are transferred to the upper half of the designated X_a register. The lower half of X_a is unaffected.

Example:

Assume (U) = 333444 222111
$(X_a)_i$ = 123456 234567

Then $(X_a)_f$ = 222111 234567

Programming Note:

For whole word transfers, (j=0) only the lower half of the (U) is transferred to X_a.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction	Mnemonic
Store X	SX

06	j	X	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of X_a are transferred to u.

Example:

Assume (X_a) = 002255 663311

Then $(U)_f$ = 002255 663311

Note: j = 00 to 15_8. If j = 16_8 or 17_8, no storage results.

	Execution Time in usec.	
Control Registers	Alternate Bank	Same Bank
N/A	1.625 (jump) .75 (no jump)	N/A

Instruction	Mnemonic
Jump Modifier Greater and Increment	JMGI

74	12	X	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U when the least significant half of the a-designated index register is greater than zero. If less than or equal to zero, the next instruction is executed.

Example:

Index register Xa is to be tested to determine whether an end-of-loop has been reached. Until then, a jump is made back to the beginning of the loop. If $(Xa)_i$ = 777776000012, then the loop is repeated ten times. The address of the beginning of the loop is U. $(Xa)_f$ = 777776777776 and the next instruction in sequence is executed.

Programming Note:

In either of the above cases, incrementation of the index register takes place. Testing of the modifier takes place prior to incrementation. Either the modifier or increment portion or both can be negative.

Instruction Sequence Control

The instruction sequence defines operations that conditionally or unconditionally alter the sequential flow of a program. These operations allow the program to enter and leave subroutines; to test specific storage locations or special indicators; to determine, externally, the next logical path to enter; and to check the status of I/O operations.

The operations defined under instruction sequence can be divided into two classes: unconditional jump instructions and conditional jump instructions. The conditional jump instructions can further be divided into three subclasses: test/jump instructions, test/skip instructions and repeat search instructions.

The unconditional jump instructions preform an operation, then transfer program control to the location specified in the u-field of the instruction.

The test/jump instructions test a designated location, either control or main storage, for a specific condition. When the condition is met, program control is transferred to the location specified by the u-field of the instruction. If the condition is not met, the next instruction is executed.

The test/skip instructions, test a known value or values against an unknown value. When the condition of the test is satisfied, the next instruction is skipped and that instruction plus one is executed. If the condition is not satisfied, the next instruction is executed.

The 14 repeat search instructions are executed the number of times specified by the repeat count, K, in the R1 register. For mask search operations, the mask is contained in control register R2. The u-field should be modified by an index register in order to obtain the next storage location to be checked. Incrementation of the index register will occur only if the h-designator is set. When a location has been found that satisfies the condition, the next instruction is skipped and that instruction plus one is executed. If R1 has decremented to zero, and the specified condition has not been met, the next instruction is executed.

Execution Time
in usec.

2.125

Instruction Mnemonic

Store Location and Jump SLJ

72	01		x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The address of the next instruction, P, is transferred to the lower half of U. Program control resumes at U+1. Program control may be returned to the normal sequence, P, by executing a jump to the effective address of the SLJ+1. This instruction also captures a relative P.

Example:

Assume (P) = 1000
(U) = 200000
$(200000)_i$ = 740400065432

Then $(200000)_f$ = 740400001000

Program control is transferred to address 200001.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Prevent all I/O interrupts and Jump	PAIJ

72	13		x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Interrupt lockout conditions are set to inhibit the honoring of all I/O interrupts. Program control is transferred to U.

Example:

Assume u = 006412

Then all I/O interrupts are temporarily inhibited and a jump is made to location 006412.

Programming Note:

Time delay of 100 usec. The Guard mode Fault occurs after this delay. This instruction is used when it is desired to anticipate I/O interrupts and to prevent them from being honored by the computer. It is used primarily in situations where the current program has time priority and it is desirable to lockout I/O interrupts until the current operation has been completed.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Allow All I/O Interrupts and Jump	AAIJ

74	07		x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Interrupt lockout conditions are cleared and program control is transferred to U.

Example:

Assume U = 006250

Then program control is transferred to address 006250 and the I/O Interrupt Lockout is cleared.

Programming Note:

This instruction must be programmed within the I/O interrupt routine; either as the return to the main program, or before performing any further I/O operations within the interrupt routine.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction — Mnemonic

Load Modifier and Jump — LMJ

74	13	X	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The storage address register, P, is automatically made program relative and stored in the designated X_a register. Program control is transferred to U.

Example:

Assume $(X_a)_i$ = 123456 222333
(P) = 100
(U) = 003000

Then $(X_a)_f$ = 123456 000100 and program control is transferred to address 3000. The return address for resumption of the main routine is available in X_a.

	Execution Time in usec.	
Control Registers	Alternate Bank	Same Bank
N/A	.75 (jump) 1.50 (no jump)	N/A

Instruction	Mnemonic
Jump On Greater and Decrement	JGD

70	j	a	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The j and a designators together specify a control register. Program control is transferred to U if the (ja) is greater than zero. The (ja) is decremented whether the jump is executed or not. If the (ja) is less than or equal to zero, program control resumes at the next instruction.

Programming Note:

The j and a-designators are combined to reference a single control register. The total value of (a+j) must not exceed 177_8 since only bit positions 28-22 are considered by the computer in interpreting this address.

	Execution Time in usec.	
Control Registers	Alternate Bank	Same Bank
N/A	1.625 (jump) .875 (no jump)	N/A

Instruction	Mnemonic
Double Precision Zero Jump	DJZ

71	16	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U if the (A, A+1) are all zeros or all ones. Control resumes at the next instruction if the (A, A+1) is not equal to zero.

Example:

Assume U = 001000
(A) = 000000000000
(A+1) = 000000000000
Then program control is transferred to address 01000.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	1.50 (jump) .75 (no jump)	N/A

Instruction	Mnemonic
Jump on Positive and Shift	JPS

72	02	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program control is transferred to U if the (A) is positive. The next instruction is executed if the (A) is negative. The (A) are shifted left circularly 1 bit position regardless of sign.

Example:

Assume U = 050000
$(A)_i$ = 345670 111222

Then $(A)_f$ = 713560 222444 and program control is transferred to address 050000.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	1.50 (jump) .75 (no jump)	N/A

Instruction	Mnemonic
Jump on Negative and Shift	JNS

72	03	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U if the (A) is negative. The next instruction is executed if the (A) is positive. The (A) are shifted left circular 1 bit position regardless of sign.

Example:

Assume U = 000355
$(A)_i$ = 500000 000000

Then $(A)_f$ = 200000 000001 and program control is transferred to address 0355.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	1.50 (jump) .75 (no jump)	N/A

Instruction	Mnemonic
Jump on Zero	JZ

74	00	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U if the (A) are all zeros or all ones. The next instruction is executed if the (A) is not equal to zero.

Example:

Assume U = 000100
(A) = 777777 777777

Then program control is transferred to address 0100.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	1.50 (jump) .75 (no jump)	N/A

Instruction	Mnemonic
Jump on Non Zero	JNZ

74	01	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U if the (A) is not equal to plus or minus zero. If the (A) is plus or minus zero the next instruction is executed.

Example:

Assume U = 005000
(A) = 321321 432432

Then program control is transferred to address 05000.

	Execution Time in usec.	
Control Registers	Alternate Bank	Same Bank
N/A	1.50 (jump) .75 (no jump)	2.25 (jump) 1.50 (no jump)

Instruction Mnemonic

Jump on Positive JP

74	02	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U when the (A) is positive. If the (A) is negative, the next instruction is executed.

Example:

Assume U = 000450
(A) = 333444 555666

Then program control is transferred to address 0450.

	Execution Time in usec.	
Control Registers	Alternate Bank	Same Bank
N/A	1.50 (jump) .75 (no jump)	N/A

Instruction Mnemonic

Jump on Negative JN

74	03	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U when the (A) is negative. If the (A) is positive, the next instruction is executed.

Example:

Assume U = 000450
(A) = 333444 555666

Since the (A) is positive, the next instruction in sequence is executed.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Jump on Keys, Jump	JK, J

74	04	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

One of the 15 special keys on the operator's console is represented by the a-designator. When the key is in the on position, a jump to U will be executed; in the off position the next instruction will be executed. If the a-designator is equal to zero, a jump to U will always be executed.

Execution:

The 4 bit a-designator is a value ranging from $1\text{-}17_8$. This value represents one of the 15 special keys on the operator's console. If the a-designator is equal to zero, program control is unconditionally transferred to U since key zero is non-existent and considered to be always on.

Example:

Assume A = 03
U = 000400

Then if the console switch #3 is in an ON condition, program control is transferred to address 0400.

Execution Time in usec.

Control Registers	Alternate Bank	Same Bank
N/A	.75	N/A

Instruction	Mnemonic
Halt on Keys and Jump	HKJ
Halt and Jump	HJ

74	05	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

A bit by bit comparison is made between the 4 Selective Stop Switches and the 4 bit locations in the a-field. Each bit location in the a-field corresponds to one stop switch. When the logical product of the bits generated by the Select Stop Switches and the corresponding bits in the a-field are non-zero, the stop will occur. If A equals zero, stop. Program control is transferred to U regardless of switch setting. When a computer stop is made, the computer is reactivated by manually pressing the start button located on the operator's console.

Example:

A variable four bit number is developed in a program and inserted into the a-designator of a HKJ instruction with U = 035722. When HKJ is subsequently executed, a stop may or may not occur depending on the comparison explained above. In either case, program control is transferred to address 035722.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	1.50 (jump) .75 (no jump)	N/A

Instruction	Mnemonic
Jump on No Low Bit	JNB

74	10	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U when the least significant bit of the (A) is zero. If the least significant bit is one, the next instruction is executed.

Programming Note:

Since the test is only on bit position 0, the condition of a positive or negative status makes no difference.

Example:

Assume U = 000100
(A) = 345670 120000

Then program control is transferred to address 0100.

	Execution Time in usec.	
Control Registers	Alternate Bank	Same Bank
N/A	1.50 (jump) .75 (no jump)	N/A

Instruction Mnemonic

Jump on Low Bit JB

74	11	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U if the least significant bit in the designated A register is one. If the least significant bit is zero, the next instruction is executed.

Example:

Assume U = 000100
(A) = 345670 120000

Then the next instruction is executed.

	Execution Time in usec.	
Control Registers	Alternate Bank	Same Bank
N/A	1.50 (jump) .75 (no jump)	N/A

Instruction Mnemonic

Jump on Overflow JO

74	14		x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U if the Overflow designator in PSR is set. If the Overflow designator is not set the next instruction is executed.

Programming Notes:

Arithmetic additions or subtractions in the accumulator of quantities with like sign which yield a value of opposite sign set the overflow designator.

The Overflow designator is cleared prior to executing an addition or subtraction operation.

The coding of the a-designator is not required.

See overflow and carry designators in processor and storage section.

	Execution Time in usec.	
Control Registers	Alternate Bank	Same Bank
N/A	1.50 (jump) .75 (no jump)	N/A

Instruction	Mnemonic
Jump on No Overflow	JNO

74	15		x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U when the overflow indicator is not set. When the overflow indicator is set the next instruction is executed.

Programming Notes:

(See Programming Notes on the Jump on Overflow instruction)

	Execution Time in usec.	
Control Registers	Alternate Bank	Same Bank
N/A	1.50 (jump) .75 (no jump)	N/A

Instruction Mnemonic

Jump on Carry JC

74	16		x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U when the carry indicator is set. If the Carry indicator is not set , the next instruction is executed.

Programming Notes:

Addition or subtraction operations in either the accumulator or index registers can cause an end-around carry to occur. On an add instruction when a carry occurs, based on the conditions stated in the table below, the carry indicator is set.

Values	Results +	Results -
(A)+(U)	Clear	Clear
(A)+(-U)	Set	Clear
(-A)+(-U)	Set	Set
(-A)+(U)	Set	Clear

	Execution Time in usec.	
Control Registers	Alternate Bank	Same Bank
N/A	1.50 (jump) .75 (no jump)	N/A

Instruction	Mnemonic
Jump on No Carry	JNC

74	17		x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U if the Carry indicator is not set. If set, the next instruction is executed.

(See Programming Notes on the Jump on carry instruction)

	Execution Time in usec.	
Control Registers	Alternate Bank	Same Bank
N/A	.75	.75

Instruction — Mnemonic

Jump on Input Channel Busy — JIC

75	02	channel #	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U when the channel, coded in the a-designator, is in the input mode. If the channel is not in the input mode, the next instruction is executed.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Jump on Output Channel Busy	JOC

75	06	channel #	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to U when the channel specified by a ⊕ CSR is in the output mode. If the channel is not in the output mode, the next instruction is executed.

Execution Time in usec.		
Control Registers	Alternate Bank	Same Bank
N/A	.75	N/A

Instruction	Mnemonic
Jump On Function in Channel	JFC

75	12	channel #	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program control is transferred to U when the channel, specified by a ⊕ CSR is in the function mode. If the channel is not in the function mode, the next instruction is executed.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.00 (skip)	2.75 (skip)
	1.25 (no skip)	2.00 (no skip)

Instruction Mnemonic

Test Even Parity TEP

44	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

A logical product is developed based on the (A) and the (U). If this product contains an even number of bits, the next instruction is skipped.

Example (in binary notation):

Assume (A) = 001010 011100 101110 101100 011010 001000
(U) = 000000 000000 000000 111111 111111 111111

Then (A)⊙(U) = 000000 000000 000000 101100 011010 001000

Since the logical product contains an odd number of bits, the next instruction will be executed.

Programming Note:

The logical product is not available to the programmer.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.00 (skip) 1.25 (no skip)	2.75 (skip) 2.00 (no skip)

Instruction	Mnemonic
Test Odd Parity	TOP

45	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

A logical product is developed based on the (A) and the (U). If the product contains an odd number of bits, the next instruction is skipped.

Example (in binary notation):

Assume (A) = 001010 011100 101110 101100 011010 001000
(U) = 000000 000000 000000 111111 111111 111111

Then (A)⊙(U) = 000000 000000 000000 101100 011010 001000

Since the logical product contains an odd number of bits, the next instruction is skipped.

Programming Note:

The logical product is not available to the programmer.

		Execution Time in usec.	
		Alternate Bank	Same Bank
		1.75 (skip) 1.00 (no skip)	2.5 (skip) 1.75 (no skip)
Instruction	Mnemonic		
Test Less or Equal to Modifier	TLEM		
Test not greater than Modifier	TNGM		

47	j	X	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The least significant half of the (X_a) is compared to the least significant portion of the (U). If the (U) is algebraically less than or equal to the (X_a), the next instruction is skipped.

Example:

Assume (U) = 000135 001234
$(X_a)_i$ = 000002 061234

Then the next instruction is skipped and $(X_a)_f$ = 000002 061236

Programming Note:

Incrementation of the index register takes place regardless of the result of the comparison. The j-designators 0 and 2 are treated the same as a j-designator of 1.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	1.625 (skip) .875 (no skip)	2.375 (skip) 1.625 (no skip)

Instruction Mnemonic

Test for Zero TZ

50	j		x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The contents of the U address are tested. If all 0's (+0) or all 1's (-0) are present, the next sequential instruction is skipped and the following instruction is executed.

Example:

Assume (U) = 000000 777000

Then the next sequential instruction is executed.

Programming Note:

The a-designator is not used by this instruction.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	1.625 (skip) .875 (no skip)	2.375 (skip) 1.625 (no skip)

Instruction Mnemonic

Test for Non-Zero TNZ

51	j	a	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are examined for a non-zero value. If a non-zero value exists, the next instruction is skipped.

Example:

Assume (U) = 000000 777000

Then the next instruction is skipped.

Programming Note:

The a-designator is not used by this instruction.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	1.625 (skip) .875 (no skip)	1.625 (skip) .875 (no skip)	2.375 (skip) 1.625 (no skip)

Instruction Mnemonic

Test for Equal TE

52	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are compared algebraically to the (A). If the values are equal, the next instruction is skipped.

Example:

Assume (U) = 000012 003642
(A) = 000000 003642

Then the next instruction is executed.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	1.625 (skip) .875 (no skip)	1.625 (skip) .875 (no skip)	2.375 (skip) 1.625 (no skip)

Instruction Mnemonic

Test for Not Equal TNE

53	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are compared algebraically to the (A). If the values are not equal, the next instruction is skipped.

Example:

Assume (U) = 000012 003642
(A) = 000000 003642

Then the next instruction is skipped.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	1.625 (skip) .875 (no skip)	1.625 (skip) .875 (no skip)	2.375 (skip) 1.625 (no skip)

Instruction	Mnemonic
Test for Less or Equal	TLE
Test for not greater	TNG

54	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are compared algebraically to the (A). If the (U) is less than or equal to the (A), the next instruction is skipped.

Example:

Assume (U) = 000062 003567
(A) - 000000 003567

Then the next instruction is executed.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	1.625 (skip) .875 (no skip)	1.625 (skip) .875 (no skip)	2.375 (skip) 1.625 (no skip)

Instruction Mnemonic

Test for Greater TG

55	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are compared algebraically to the (A). If the (U) is greater than the (A), the next instruction is skipped.

Example:

Assume (U) = 000062 003567
(A) = 000000 003567

Then the next instruction is skipped.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	1.625 (skip) .875 (no skip)	1.75 (skip) 1.00 (no skip)	2.5 (skip) 1.75 (no skip)

Instruction Mnemonic

Test for Within Range TW

56	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are compared algebraically to be greater than the (A) and less than or equal to the (A+1). If the (U) fall within the range of (A) and (A+1), the next instruction is skipped.

Example:

Assume (A) = 777777 776543
(A+1) = 000000 003456
(U) = 777537 123456

Then the next instruction is executed.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	1.75 (skip) 1.00 (no skip)	2.5 (skip) 1.75 (no skip)

Instruction	Mnemonic
Test for Not Within Range	TNW

57	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are compared algebraically to be less than or equal to the (A) and greater than the (A+1). If the (U) fall outside the range of (A) and (A+1), the next instruction is skipped.

Example:

Assume (A) = 000000 001262
(A+1) = 000000 002253
(U) = 000122 531262

Then the next instruction is skipped.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	1.50 (skip) .75 (no skip)	1.50 (skip) .75 (no skip)	2.25 (skip) 1.50 (no skip)

Instruction Mnemonic

Test for Positive TP

60	j		x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

If the sign bit of the whole or partial word in the address designated by U is a "0", the value is positive and the next instruction is skipped.

Example:

Assume (U) = 300000 007777

Then the next instruction is skipped.

Programming Note:

The a-designator is not used by this instruction.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	1.50 (skip) .75 (no skip)	1.50 (skip) .75 (no skip)	2.25 (skip) 1.50 (no skip)

Instruction Mnemonic

Test for Negative TN

61	j		x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

If the sign bit of the whole or partial word in the address designated by U is a "1", the value is negative and the next instruction is skipped.

Example:

Assume (U) = 465432 004000

Then the next instruction is skipped.

Programming Note:

The a-designator is not used by this instruction.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.375 (skip) 1.625 (no skip)	3.125 (skip) 2.375 (no skip)

Instruction	Mnemonic
Double Precision Test Equal	DTE

71	17	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U, U+1) are compared algebraically to the (A, A+1). If the values are equal, the next instruction is skipped.

Example:

Assume (U, U+1) = 123456 000000 112233 445566
(A, A+1) = 444555 000000 123456 000000

Then the next instruction is executed.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction Mnemonic

Search for Equal SE

62	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are algebraically compared to the (A). If the (U) and the (A) are equal, the next instruction is skipped.

Example:

Assume (A) = 000000 062454
(U) = 006250
(X) = 000001 000000
(R1) = 000000 000144

Then the contents of addresses 006250 to 006414 will be compared to the (A). As soon as a value equal to 062454 has been found, the next instruction is skipped. If no match can be found, the next instruction is executed.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction	Mnemonic
Search for Not Equal	SNE

63	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are algebraically compared to the (A). When the (U) and (A) are not equal, the next instruction is skipped.

Example:

Assume (A) = 000000 006000
(U) = 006250
(X) = 000002 000000
(R1) = 000000 000062

Then the contents of every other address from 006250 to 006414 will be compared to the (A). As soon as a value not equal to 006000 has been found, the next instruction is skipped. If no match can be found, the next instruction is executed.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction	Mnemonic
Search for Less or Equal	SLE
Search for Not Greater	SNG

64	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are algebraically compared to the (A). If the (U) is less than or equal to the (A), the next instruction is skipped.

Example:

Assume (A) = 000000 000537
(X) = 777776 000144
u = 006250
(R1) = 000000 000144

Then the contents of addresses 006414 to 006250 will be compared in descending sequence to the (A). If a value less than or equal to 000537 is found, the next instruction is executed.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction Mnemonic

Search for Greater SG

65	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are algebraically compared to the (A). When the (U) are greater than the (A), the next instruction is skipped.

Example:

The example for the Search for Less or Equal instruction pertains, except the next instruction is skipped if a value greater than (A) is found. If no value can be found to satisfy this condition, the next instruction is executed.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction Mnemonic

Search for Within Range SW

66	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are algebraically compared to the (A, A+1). If the (U) are greater than the (A) and less than or equal to the (A+1), the next instruction is skipped.

Example:

```
Assume  (U)   = 006250
        (A)   = 000000 010000
      (A+1)   = 000000 020000
       (R1)   = 000000 000144
        (X)   = 000001 000000
```

Then the contents of each address between 006250 and 006414 will be tested to see if it is greater than 010000, but less than or equal to 020000. If such a value is found, the next instruction is skipped. If no such value is found, the next instruction is executed.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction	Mnemonic
Search for Not Within Range	SNW

67	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are algebraically compared to the (A, A+1). If the (U) are less than or equal to the (A) or greater than the (A+1), the next instruction is skipped.

Example:

```
Assume  u     = 006250
       (A)    = 000000 010000
     (A+1)    = 000000 020000
     (R1)     = 000000 000144
       (X)    = 000001 000000
```

Then the contents of each address between 006250 and 006414 will be to see if it is less than or equal to 010000, or greater than 020000. If such a value is found, the next instruction is skipped. If no such value is found, the next instruction is executed.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction	Mnemonic
Masked Search for Equal	MSE

71	00	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

A logical product is developed between the mask in R2 and the (A). For each repeat cycle a logical product is also formed between R2 and the (U), this result is algebraically compared to the previously developed product of the (R2) and the (A). If the compared values are equal, the next instruction is skipped.

Example:

Assume $(R1)_i$ = 000000 000001
(A) = 253012 333403
(U) = 253444 123457
(R2) = 777000 000001

Then (R2) $\odot$ (A) = 253000 000001
and (R2) $\odot$ (U) = 253000 000001

Since these values are equal, the next instruction is skipped.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction	Mnemonic
Masked Search for Not Equal	MSNE

71	01	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

A logical product is developed between the mask in R2 and the (A). For each repeat cycle, a logical product is also formed between R2 and the (U), this result is then compared to the previously developed product of R2 and the (A). If the compared values are not equal, the next instruction is skipped.

Example:

Assume $(R1)_i$ = 000000 000001
(A) = 253012 333403
(U) = 253444 123457
(R2) = 777000 000001

Then (R2) ⊙ (A) = 253000 000001
and (R2) ⊙ (U) = 253000 000001

Since these values are equal and the repeat count in R1 is now zero, the next instruction is executed.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction	Mnemonic
Masked Search for Less or Equal	MSLE
Masked Search for Not Greater	MSNG

71	02	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

A logical product is developed between the mask in R2 and the (A). For each repeat cycle a logical product is also formed between R2 and the (U). This result is algebraically compared to the previously developed product of R2 and A. If the product of R2 and the (U) is less than or equal to the product of R2 and the (A), the next instruction is skipped.

Example:

Assume	(R1)	=	000000 000001
	(A)	=	123456 012345
	(U)	=	044444 012034
	(R2)	=	000000 000077
then	(R2) ⊙ (A)	=	000000 000045
and	(R2) ⊙ (U)	=	000000 000034

Since 034 is less than 045, the next instruction is skipped.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction — Mnemonic

Masked Search for Greater — MSG

71	03	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

A logical product is developed between the mask in R2 and the (A). For each repeat cycle a logical product is also formed between R2 and the (U). This result is algebraically compared to the previously developed product of R2 and the (A). If the product of R2 and the (U) is greater than the product of R2 and the (A), the next instruction is skipped.

Example:

Assume	(R1)	=	000000 000001
	(A)	=	123456 012345
	(U)	=	044444 012034
	(R2)	=	000000 000077
then (R2) ⊙ (A)		=	000000 000045
and (R2) ⊙ (U)		=	000000 000034

Since 034 is not greater than 045 and the repeat count in R1 is down to zero, the next instruction is executed.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction — Mnemonic

Masked Search for Within Range — MSW

71	04	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

A logical product is developed for the lower limit, between the mask in R2 and the (A), and the upper limit between R2 and the (A+1). For each repeat cycle a logical product is also formed between R2 and the (U). This result is algebraically compared to the developed products of R2 and the (A), and R2 and the (A+1). If the product of R2 and the (U) falls within the range, the next instruction is skipped.

Example:

The example under the Search Within Limits instruction would operate in the same way except that both the (A) and the (U) are modified by a logical multiplication by the mask in R2 before the testing is done.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction — Mnemonic

Masked Search for Not Within Range — MSNW

71	05	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

A logical product is developed for the lower limit between the mask in R2 and the (A), and for the upper limit between R2 and the (A+1). For each repeat cycle a logical product is also formed between R2 and the (U). This result is algebraically compared to the developed products of R2 and the (A), and R2 and the (A+1). If the product of R2 and the (U) falls outside the range, the next instruction is skipped.

Example:

The example under the Search for Not Within Range instruction would operate in the same way except that both the (A) and the (U) are modified by a logical multiplication by the mask in R2 before the testing is done.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction — Mnemonic

Masked Alphanumeric Search for Less or Equal — MASL

71	06	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

A logical product is developed between the mask in R2 and the (A). For each repeat cycle a logical product is also formed between R2 and the (U). This product is compared bit by bit to the previously developed product of R2 and the (A). If the product of R2 and the (U) is less than or equal to the product of R2 and the (A), the next instruction is skipped.

Programming Note-

The masked portions of the (A) and the (U) are compared bit by bit and not algebraically as in the previous mask search operations.

	Execution Time in usec.	
	Alternate Bank	Same Bank
	2.25 + .75K	3.0 + .75K

Instruction Mnemonic

Masked Alphanumeric Search for Greater MASG

71	07	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

A logical product is developed between the mask in R2 and the (A). For each repeat cycle a logical product is also formed between R2 and the (U). This product is compared bit by bit to the previously developed product of R2 and the (A). If the product of R2 and the (U) is greater than the product of R2 and the (A), the next instruction is skipped.

Programming Note:

The masked portions of the (A) and the (U) are compared bit by bit and not algebraically as in the previous mask search operations.

Executive Control

The Executive Control Instructions are defined as a set of operations which enable an executive routine to maintain proper control over worker program execution.

	Execution Time in usec.	
Control Registers	Alternate Bank	Same Bank
N/A	1.375	N/A

Instruction	Mnemonic
Executive Return	ER

72	11	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Program Control is transferred to the Executive Return Interrupt location.

Programming Note:

The execution of this instruction generates an internal interrupt to location 242_8. It also triggers the storing of the operating program's (PSR) Internal Function Register into Index Register, 0. (Processor State Word).

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Store Channel Number SCN

72	14		x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Upon the generation of an interrupt, this instruction, will store the appropriate channel number at the location specified by the (U).

Programming Note:

When the channel number is stored in main storage, bit locations 3-0 contain the channel number, bit locations 17-4 are always zero, and bit locations 35-18 are zero.

When the channel number is stored in a control register, bit locations 3-0 contain the channel number and bit locations 35-4 are unaffected.

When the u-field is coded the a-designator is not used.

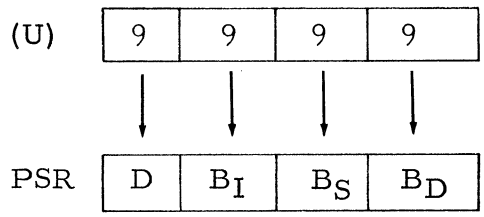
	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction	Mnemonic
Load Processor State Register	LPS

72	15	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are transferred to the Processor State Register.



Programming Notes:

1. This instruction is in effect two instructions later.
2. It must be followed by either a JP or NO-OP instruction.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction — Mnemonic

Load Storage Limits — LSL

72	16	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are transferred to the Storage Limits Register.

(U)	9	9	9	9
	↓	↓	↓	↓
	upper limit I area	lower limit I area	upper limit D area	lower limit D area

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
Initiate Interprocessor (Synchronous) Interrupt	III

73	14	A				
35 30	29 26	25 22	21 18	17	16	15 00

Description:

An interrupt is sent over one of two external synchronous lines to the connected device. When the a-designator is coded with a zero, the interrupt is sent on line 0. If coded with a one, the interrupt is sent on line 1.

Programming Note:

If the a-designator is coded with a ten, the program alarm on the operator's console sounds. The x-, h-, i-, and u-fields are not to be coded.

	Execution Time in usec.	
Control Registers	Alternate Bank	Same Bank
N/A	.75	N/A

Instruction Mnemonic

Select Interrupt Location SIL

73	15	A				u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The storage interrupt address assignment is moved to the storage module as coded in the a-field.

a-field	Storage Module	absolute address
00_2	1	$200\text{-}247_8$
01_2	2	$100200\text{-}100247_8$
10_2	3	$200200\text{-}200247_8$
11_2	4	$300200\text{-}300247_8$

Programming Note:

1. The x-, h-, i- and u-fields are not to be coded.
2. This instruction will allow shutdown of one or more modules in case of fault.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.875	.875	1.625

Instruction Mnemonic

Load Channel Select Register LCR

73	16	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The Channel Select Register (CSR) is loaded with the (U), bit positions 3-Ø.

Programming Note:

The channel number is computed by the logical addition of the a-designator and the (CSR). When the CSR is cleared to zero, the I/O section obtains the channel number from the a-designator. This is for I/O instructions requiring channels.

Special Purpose

The instructions that have not been defined previously are explained in this section. They do not lend themselves to any particular group or to each other.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction	Mnemonic
No Operation	NOP

74	06	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Upon execution of this instruction there is a delay of 750 nanoseconds after which the next instruction is executed.

1. Indexing is effective on this instruction.
   (If h = 1, the specified index register is incremented.)

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	N/A	.75	N/A

Instruction Mnemonic

Execute EX

72	10		x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The instruction at the location U is executed. The P register is increased only when the last step of the instruction stored at U is completed, the instruction sequence resumes at the address following the Execute.

Programming Note-

If the remote address contains a skip or jump instruction, the normal sequence is interrupted and return to the instruction following the Execute must be programmed.

Input/Output

The Input/Output operations enable a program to communicate between the processor and the peripheral equipment.

Each of the sixteen channels are capable of operating in three different modes: Function, Input and Output. The Function Mode establishes the initial communication between the processor and the peripheral subsystem. The Input/Output Modes allow the transferring of data between the processor and the peripheral units.

Associated with each channel are two Access Control Register's, the Input Access Control Registers occupying control register addresses 40_8-57_8. The Output Access Control Registers, control register addresses 60_8-77_8, regulate data coming from the processor. The relationship between the channel number and the appropriate control register location is, in the case of input, to add the channel number to the base address of the Input Access Control Register, 40_8. Thus channel 0 corresponds to control register address 40_8, channel 1 to control memory address 41_8, etc. The above rule also applies to the Output Access Control Registers with the exception that the base address now becomes 60_8.

The format for the Access Control Word is given below:

G W V

V = initial storage address to or from which data will be transferred.
W = indicates the number of words to be transferred.
G = increment designator
= 00-Increase the V portion of the Access Control Words by 1 each time the word is referenced for a transfer.
= 10 Decrease the V portion of the Access Control Words by 1 each time the word is referenced for a transfer.
= 01 Does not increase or decrease the V portion of the Access 11 Control Word.

The basic I/O instruction Word consists of 7 designators each represented by 2 letters. (See section 3 for details on the Instruction Word.)

f	j	a	x	h	i	u

In the execution of an I/O instruction, the j-designator combines with the f-designator to form a 10 bit function code. The a-designator specifies the I/O

channel. The x, h, and i-designators determine index modification, incrementation, and indirect addressing. The u-designator specifies the address of the Access Control Word. When the I/O instruction word is decoded, it gives the mode of operation, activates the selected channel, initiates the transfer, loads the Access Control Register, and conditions the I/O section for transfer. From the time of initiation, the data transfer is controlled by the Access Control Word. If no error or malfunction occurs, the data transfer will continue until the count in the W-portion of the Access Word reaches zero, or until end-of-tape is detected; an external interrupt is then issued. When the count reaches zero, a signal is generated to end the data transfer and to interrupt instruction execution, if programmed.

	Execution Time in usec.	
	Control Registers	Alternate Bank
	N/A	.75

Instruction: Disconnect Input Channel — Mnemonic: DIC

75	03	C	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Terminates the input mode on the channel specified by a ⊕ CSR coded in the C a-designator.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Load Input Channel LIC

75	00	C	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are transferred to the appropriate Input Access Control Register. The function mode is then initiated on the channel specified by a ⊕ CSR.

Programming Note:

This instruction must be programmed prior to the Load Function Channel (75-10 or 11).

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction — Mnemonic

Load Input Channel and Monitor — LICM

75	01	C	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are transferred to the appropriate Input Access Control Register. The function mode is then initiated on the channel specified by a ⊕ CSR. Upon completion of the transfer a monitor interrupt will be generated.

Programming Note:

The instruction must be programmed prior to the Load Function Channel (75-10 or 11).

Execution Time in usec.

Control Register	Alternate Bank	Same Bank
N/A	.75	.75

Instruction	Mnemonic
Disconnect Output Channel	DOC

75	07	C	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

Terminates the output or function mode on the specified channel coded in the C a-designator.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Load Output Channel LOC

75	04	C	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are transferred to the appropriate Output Access Control Register. The output mode is initiated on the channel specified by a ⊕ CSR.

Programming Note:

This instruction is usually programmed after the Load Function Channel (75-10,11).

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction — Mnemonic

Load Output Channel and Monitor — LOCM

75	05	C	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are transferred to the appropriate Output Access Control Register. The output mode is initiated on the channel specified by a ⊕ CSR. Upon completion of the transfer a monitor interrupt is generated.

Programming Note:

This instruction is usually programmed after the Load Function Channel (75-10,11).

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Load Function in Channel LFC

75	10	C	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are transferred to the appropriate Output Access Control Register. The function mode is then initiated on the channel specified by a ⊕ CSR.

Programming Note:

This instruction must be programmed prior to the Load Output Channel (75-04,05) and the following the Load Input Channel (75-00,01).

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Load Function in Channel and Monitor LFCM

75	11	C	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are transferred to the appropriate Output Access Control Register. The function mode is then initiated on the channel specified by a ⊕ CSR. Upon completion of the transfer a monitor interrupt is generated.

Programming Note:

This instruction must be programmed prior to the Load Output Channel (75-04, 05) and following the Load Input Channel (75-00, 01).

		Execution Time in usec.		
		Control Registers	Alternate Bank	Same Bank
		N/A	.75	.75
Instruction	Mnemonic			
Allow all External Interrupts	AACI			

75	14	C	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

This instruction provides for enabling all External Interrupts on all I/O channels.

Execution Time
in usec.

Control Registers	Alternate Bank	Same Bank
N/A	.75	.75

Instruction	Mnemonic
Prevent All External Interrupt	PACI

75	15	C	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

This instruction provides for disabling all External Interrupts on all I/O channels. It is possible to get one more External Interrupt after programming the PACI (75-15). This is because all External Interrupts are prevented from coming into the computer, but an External Interrupt may already be in the computer's I/O hardware section. This is a physical characteristic of the I/O hardware and cannot be eliminated by software manipulation. If an External Interrupt is already in the I/O hardware, and the PACI (75-15) is programmed, the one already in the computer's I/O hardware will be honored.

Logical Operations

The 4 logical instruction perform a logical addition, subtraction or multiplication. The result of the particular operation is stored in A+1. The original (A) and (U) are unchanged.

With the exception of the Mask Load Upper, all other operations develop a logical result based on the mask which is referenced by the (U) and the designated A register.

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction	Mnemonic
Logical OR	OR

40	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are compared bit by bit with the (A) to form a logical sum in A+1. For each one bit in the (U), a one bit replaces a zero or one in the corresponding bit position in A.

Example (in binary notation):

Assume (A) = 010101 010101 010101 010101 010101 010101
(U) = 000000 000000 111111 110011 000000 111111

Then $(A+1)_f$ = 010101 010101 111111 110111 010101 111111

Programming Note:

The logical sum is formed based on the following truth table:

(A)	⊕	(U)	→	A+1
0	⊕	0	=	0
0	⊕	1	=	1
1	⊕	1	=	1
1	⊕	0	=	1

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Logical Exclusive OR XOR

41	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

The (U) are compared bit by bit with the (A). For each one bit in the (U), complement the corresponding bit position in A. The logical difference is stored in A+1.

Programming Note:

The logical difference is formed based on the following truth table:

(A)	$\overline{\oplus}$	(U)	→	A+1
0	$\overline{\oplus}$	0	=	0
0	$\overline{\oplus}$	1	=	1
1	$\overline{\oplus}$	0	=	1
1	$\overline{\oplus}$	1	=	0

Example (in binary notation):

```
Assume (A)   = 010101 010101 010101 010101 010101 010101
       (U)   = 000000 000000 111111 110011 000000 111111
               ------------------------------------------
Then (A+1)f  = 010101 010101 101010 100110 010101 101010
```

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Logical AND AND

42	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

For each zero bit in the (U), the corresponding bit position in A is cleared to zero. For each one bit in the (U), the corresponding zero or one bit in A is retained. The logical product is stored in A+1.

Example (in binary notation):

```
Assume (A)   = 001010 011100 101110 110101 100011 010001
       (U)   = 000000 000000 111111 111111 000000 000000
               ------------------------------------------
Then (A+1)f  = 000000 000000 101110 110101 000000 000000
```

Programming Note:

The logical product is formed based on the following truth table:

(A)	⊙	(U)		A+1
0	⊙	0	=	0
0	⊙	1	=	0
1	⊙	0	=	0
1	⊙	1	=	1

	Execution Time in usec.		
	Control Registers	Alternate Bank	Same Bank
	.75	.75	1.50

Instruction Mnemonic

Masked Load Upper MLU

43	j	A	x	h	i	u
35 30	29 26	25 22	21 18	17	16	15 00

Description:

A logical product is developed separately between the mask in R2 and the (U); and between the complement of the mask in R2 and the (A). The two products are algebraically added and stored in A+1.

Example (in binary notation):

Assume	(A)	=	101010	101010	000110	110010	101010	101010
	(U)	=	001100	111001	100010	001111	100011	001010
	(R2)	=	111111	111111	000000	000000	111111	111111
Then	(R2)' ⊙ (A)	=	000000	000000	000110	110010	000000	000000
	(R2) ⊙ (U)	=	001100	111001	000000	000000	100011	001010
	$(A+1)_f$	=	001100	111001	000110	110010	100011	001010

APPENDIX A. CODE/SYMBOL RELATIONSHIPS

The following table shows the relationships between the octal computer codes, the 80 column card codes, and the characters or symbols represented by these codes.

COMPUTER CODE (OCTAL)	CARD CODE	CHARACTER	COMPUTER CODE (OCTAL)	CARD CODE	CHARACTER
00	7-8	@	40	12-4-8	)
01	12-5-8	[	41	11	—
02	11-5-8	]	42	12	+
03	12-7-8	#	43	12-6-8	<
04	11-7-8	Δ	44	3-8	=
05	(Blank)	(Space)	45	6-8	>
06	12-1	A	46	2-8	&
07	12-2	B	47	11-3-8	$
10	12-3	C	50	11-4-8	*
11	12-4	D	51	0-4-8	(
12	12-5	E	52	0-5-8	%
13	12-6	F	53	5-8	:
14	12-7	G	54	12-0	?
15	12-8	H	55	11-0	!
16	12-9	I	56	0-3-8	, (comma)
17	11-1	J	57	0-6-8	\
20	11-2	K	60	0	0
21	11-3	L	61	1	1
22	11-4	M	62	2	2
23	11-5	N	63	3	3
24	11-6	O	64	4	4
25	11-7	P	65	5	5
26	11-8	Q	66	6	6
27	11-9	R	67	7	7
30	0-2	S	70	8	8
31	0-3	T	71	9	9
32	0-4	U	72	4-8	' (apostrophe)
33	0-5	V	73	11-6-8	;
34	0-6	W	74	0-1	/
35	0-7	X	75	12-3-8	.
36	0-8	Y	76	0-7-8	☐
37	0-9	Z	77	0-2-8	≠ (or stop)

APPENDIX B. INSTRUCTION REPERTOIRE (By Function Code)

Instruction Code f	Instruction Code j	Mnemonic	Instruction	Description	Execution Time + in usec.
00	–	–	Illegal Code		–
01	0-15	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	SR	Store R	(R_a)→U	.75
05	0-15	SZ	Store Zero	Zeroes→U	.75
06	0-15	SX	Store X	(X_a)→U	.75
07	–	–	Illegal Code		–
10	0-17	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	AA	Add to A	(A)+(U)→A	.75
15	0-17	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	0-17	BT	Block Transfer, repeat	$(X_x+u)\rightarrow(X_a+u)$, repeat	1.50+1.5K **
23	0-17	LR	Load R	(U)→R_a	.75
24	0-17	AX	Add to X	(X_a)+(U)→X_a	.75
25	0-17	ANX	Add Negative to X	(X_a)−(U)→X_a	.75
26	0-17	LXM	Load X Modifier	(U)→$X_{a_{17-0}}$	.875
27	0-17	LX	Load X	(U)→X_a	.75
30	0-17	MI	Multiply Integer	(A)·(U)→A, A+1	2.375
31	0-17	MSI	Multiply Single Integer	(A)·(U)→A	2.375
32	0-17	MF	Multiply Fractional	(A)·(U)→A, A+1	2.375
33	–	–	Illegal Code		–

** *For definition of K see note at end of table.*

Instruction Code f	Instruction Code j	Mnemonic	Instruction	Description	Execution Time + in usec.
34	0-17	DI	Divide Integer	(A,A+1) ÷(U)→A, Remainder→A+1	10.125
35	0-17	DSF	Divide Single Fractional	(A)÷(U)→A+1	10.125
36	0-17	DF	Divide Fractional	(A,A+1)÷(U)→A, Remainder→A+1	
37	–	–	Illegal Code		–
40	0-17	OR	Logical OR	(A)⊕(U)→A+1	.75
41	0-17	XOR	Logical Exclusive OR	(A)$\overline{\oplus}$(U)→A+1	.75
42	0-17	AND	Logical AND	(A)⊙(U)→A+1	.75
43	0-17	MLU	Masked Load Upper	(U)⊙(M)+A⊙(M)$'$→A+1	.75
44	0-17	TEP	Test Even Parity	Skip if (A)⊙(U) is even parity	2.00/1.25
45	0-17	TOP	Test Odd Parity	Skip if (A)⊙(U) is odd parity	2.00/1.25
46	0-17	LXI	Load X Increment	(U)→$X_{a_{35-18}}$	1.00
47	0-17	TLEM TNGM	Test Less or Equal to Modifier Test Not Greater than Modifier	Skip if $(X_a)_{17-0} \geq$ (U)	1.75/1.00
50	0-17	TZ	Test for Zero	Skip if (U)≠±0	1.625/ .875
51	0-17	TNZ	Test for Non Zero	Skip if (U)≠±0	1.625/ .875
52	0-17	TE	Test for Equal	Skip if (A)=(U)	1.625/ .875
53	0-17	TNE	Test for Not Equal	Skip if (A)≠(U)	1.625/ .875
54	0-17	TLE TNG	Test for Less or Equal Test for Not Greater	Skip if (U)≤(A)	1.625/ .875
55	0-17	TG	Test for Greater	Skip if (U)>(A)	1.625/ .875
56	0-17	TW	Test for Within Range	Skip if (A)<(U)≤(A+1)	1.75 /1.00
57	0-17	TNW	Test for Not Within Range	Skip if (U)≤(A) or (U)>(A+1)	1.75 /1.00
60	0-17	TP	Test for Positive	Skip if $(U)_{35}=0$	1.50 / .75
61	0-17	TN	Test for Negative	Skip if $(U)_{35}=1$	1.50 / .75
62	0-17	SE	Search for Equal	Skip if (U)=A, repeat	2.25+.75K
63	0-17	SNE	Search for Not Equal	Skip if (U)≠ A, repeat	2.25+.75K
64	0-17	SLE SNG	Search for Less or Equal Search for Not Greater	Skip if (U)≤(A), repeat	2.25+.75K
65	0-17	SG	Search for Greater	Skip if (U)>(A), repeat	2.25+.75K

Instruction Code f	Instruction Code j	Mnemonic	Instruction	Description	Execution Time + * in usec.
66	0-17	SW	Search for Within Range	Skip if (A)<(U)≤(A+1), repeat	2.25+.75K**
67	0-17	SNW	Search for Not Within Range	Skip if (U)≤(A) or (U)>(A+1), repeat	2.25+.75K
70	†	JGD	Jump on Greater and Decrement	Jump to U if (JA)>0, then (JA)−1→JA	.75 / 1.50
71	00	MSE	Mask Search for Equal	Skip if (U)⊙(M)=(A)⊙(M), repeat	2.25+.75K
71	01	MSNE	Mask Search for Not Equal	Skip if (U)⊙(M)≠(A)⊙(M), repeat	2.25+.75K
71	02	MSLE	Mask Search for Less or Equal	Skip if (U)⊙(M)≤(A)⊙(M), repeat	2.25+.75K
		MSNG	Mask Search for Not Greater		
71	03	MSG	Mask Search for Greater	Skip if (U)⊙(M)>(A)⊙(M), repeat	2.25+.75K
71	04	MSW	Masked Search for Within Range	Skip if (A)⊙(M)<(U)⊙(M) ≤(A+1)⊙(M), repeat	2.25+.75K
71	05	MSNW	Masked Search for Not Within Range	Skip if (U)⊙(M)≤(A)⊙(M) or (U)⊙(M)>(A+1)⊙(M), repeat	2.25+.75K
71	06	MASL	Masked Alphanumeric Search for Less or Equal	Skip if (U)⊙(M)≤(A)⊙(M), repeat	2.25+.75K
71	07	MASG	Masked Alphanumeric Search for Greater	Skip if (U)⊙(M)>(A)⊙(M), repeat	2.25+.75K
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 Zero Jump	Jump to U if (A,A+1)=±0	.875/1.625
71	17	DTE	Double Precision Test Equal	Skip if (U,U+1)=(A,A+1)	2.375/1.625
72	00		Illegal Code		–
72	01	SLJ	Store Location and Jump	$(P)-B_j \rightarrow U_{17-0}$, jump to U+1	2.125
72	02	JPS	Jump on Positive and Shift	if $(A)_{35}=0$, jump to U and shift (A) left one position	1.50 / .75
72	03	JNS	Jump on Negative and Shift	if $(A)_{35}=1$, jump to U and shift (A) left one position	1.50 / .75
72	04	AH	Add Halves	$(A)_{17-0}+(U)_{17-0} \rightarrow (A)_{17-0}$, $(A)_{35-18}+(U)_{35-18} \rightarrow A_{35-18}$	.75

† *The j and a designators together serve to specify any of the 128 control registers.*

** *For definition of K see note at end of table.*

Instruction Code f	Instruction Code j	Mnemonic	Instruction	Description	Execution Time + in usec.
72	05	ANH	Add Negative Halves	$(A)_{17-0} - (U)_{17-0} \rightarrow A_{17-0}$ $(A)_{35-18} - (U)_{35-18} \rightarrow A_{35-18}$	.75
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
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
72	10	EX	Execute	Execute the instruction at U	.75
72	11	ER	Executive Return	(PSR)→000,000 Exec Mode and Exec A+R mode set, Guard Mode off, program lockin off, interrupt to location $162_{10} = 242_8$	1.375
72	12		Illegal Code		
72	13	PAIJ	Prevent All I/O Interrupts and Jump	Disable all I/O Interrupts and Jump to U	.75
72	14	SCN	Store Channel Number	Channel Number → U †	.75
72	15	LPS	Load Processor State Register	(U) → PSR	.75
72	16	LSL	Load Storage Limits Register	(U)→ Storage Limits Register	.75
72	17		Illegal Code		–
73	00	SSC	Single Shift Circular	Shift (A) right U places circularly	.75
73	01	DSC	Double Shift Circular	Shift (A,A+1), right U places circularly	.75
73	02	SSL	Single Shift Logical	Shift (A) right U places, fill zeros	.75
73	03	DSL	Double Shift Logical	Shift (A,A+1) right U places, fill zeros	.75
73	04	SSA	Single Shift Algebraic	Shift (A) right U places, sign fill	.75
73	05	DSA	Double Shift Algebraic	Shift (A,A+1) right U places, sign fill	.75
73	06	LSC	Load Shift and Count	(U) → A, Shift A left circularly until $A_{35} \neq A_{34}$. If (u, u+1) is all zeros or all ones, a shift count of 35 results. Store the result in A and the number of shifts in A+1.	1.125
73	07	DLSC	Double Load Shift and Count	Same as single shift except (U,U+1)→A,A+1, result in A,A+1, shift count in A+2. If (u, u+1) is all zeros, or all ones, a shift count of 71 results.	2.125

† U_{17-0} if main storage

U_{35-0} if control register

Instruction Code f	Instruction Code j	Mnemonic	Instruction	Description	Execution Time + in usec.
73	10	LSSC	Left Single Shift Circularly	Shift (A) left U places, circularly	.75
73	11	LDSC	Left Double Shift Circularly	Shift (A,A+1 left U places, circularly	.75
73	12	LSSL	Left Single Shift Logical	Shift (A) left U places, fill zeros	.75
73	13	LDSL	Left Double Shift Logical	Shift (A,A+1) left U places, fill zeros	
73	14	III	Initiate Inter-Processor Interrupt	Initiate inter-processor interrupt as specified by the a designator	.75
73	15	SIL	Select Interrupt Module	If A = 0, assign interrupt locations to storage module 1; if A = 1, assign interrupt locations to storage module 2 if A = 2, assign interrupt locations to storage module 3 if A = 3, assign interrupt locations to storage module 4	.75
73	16	LCR	Load Channel Select Register	$(U)_{3-0} \rightarrow$ CSR	.875
74	00	JZ	Jump on Zero	If(A) = ±0, jump to U	1.50 / .75
74	01	JNZ	Jump on Non Zero	If(A) ≠ 0, jump to U	1.50 / .75
74	02	JP	Jump on Positive	If$(A)_{35}$ = 0, jump to U	1.50 / .75
74	03	JN	Jump on Negative	If$(A)_{35}$ = 1, jump to U	1.50 / .75
74	04	JK J	Jump on Keys Jump	If A = key which is set, or if A = 0 jump to U	.75
74	05	HKJ HJ	Halt on Keys and Jump Halt and Jump	Stop if A = 0, or if A ⊙ key setting ≠ 0	
74	06	NOP	No Operation	Proceed to next instruction	.75
74	07	AAIJ	Allow all Interrupts and Jump	Enable all interrupts and Jump to U	.75
74	10	JNB	Jump on No Low Bit	If$(A)_0$ = 0, jump to U	1.50 / .75
74	11	JB	Jump on Low Bit	If$(A)_0$ = 1, jump to U	1.50 / .75
74	12	JMGI	Jump Modifier Greater and Increment	If$(X_a)_{17-0}$ > 0, jump to U	1.625/ .75
74	13	LMJ	Load Modifier and Jump	$(P) \rightarrow X_{a17-0}$, jump to U	.75
74	14	JO	Jump on Overflow	Jump to U if o.f. designator set	.75 / 1.50
74	15	JNO	Jump on No Overflow	Jump to U if o.f. designator not set	.75 / 1.50
74	16	JC	Jump on Carry	Jump to U if carry designator set	.75 / 1.50
74	17	JNC	Jump on No Carry	Jump to U if carry designator not set	.75 / 1.50

Instruction Code f	Instruction Code j	Mnemonic	Instruction	Description	Execution Time + in usec.
75	00	LIC	Load Input Channel	(U) → input control word a⊕CSR, initiate input mode on channel a⊕CSR	.75
75	01	LICM	Load Input Channel and Monitor	(U) → input control word a⊕CSR, initiate input mode on channel a⊕CSR with monitor	.75
75	02	JIC	Jump on Input Channel Busy	If channel a⊕CSR in input mode, jump to U	.75
75	03	DIC	Disconnect input Channel	Terminate input mode on channel a⊕CSR	.75
75	04	LOC	Load Output Channel	(U) → output control word a⊕CSR, initiate output mode on channel a⊕CSR	.75
75	05	LOCM	Load Output Channel and Monitor	(U) → output control word a⊕CSR, initiate output mode on channel a⊕CSR with monitor	.75
75	06	JOC	Jump on Output Channel Busy	If channel a⊕CSR is in output mode, jump to U	.75
75	07	DOC	Disconnect Output Channel	Terminate output mode on channel a⊕CSR	.75
75	10	LFC	Load Function in Channel	(U) → output control word a⊕CSR, and initiate Function mode on channel a⊕CSR	.75
75	11	LFCM	Load Function in Channel	(U) → output control word a⊕CSR, and initiate Function mode on channel a⊕CSR with monitor	.75
75	12	JFC	Jump on Function in Channel	If channel a⊕CSR is in function mode, jump to U	.75
75	13		Illegal Code		
75	14	AACI	Allow All External Interrupts	All external interrupts are allowed	.75
75	15	PACI	Prevent All External Interrupts	All external interrupts are disabled	.75
75	16		Illegal Code	If Guard Mode is set, causes Guard Mode interrupt to address 243_8.	
75	17		Illegal Code		
76	00	FA	Floating Add	(A) + (U) → A,A+1	1.75
76	01	FAN	Floating Add Negative	(A) − (U) → A,A+1	1.75
76	02	FM	Floating Multiply	(A) · (U) → A,A+1	2.625

Instruction Code f	Instruction Code j	Mnemonic	Instruction	Description	Execution Time + * in usec.
76	03	FD	Floating Divide	(A) ÷ (U) → A, Remainder A+1	8.00 / 8.25
76	04	LUF	Load and Unpack Floating	Unpack (U), store fixed-point part in A+1, and store biased exponent in A_{7-0}	.75
76	05	LCF	Load and Convert to Floating	Standardize (U), pack with biased exponent from (A), and store at A+1	1.125
76	06	MCDU	Magnitude of Characteristic Difference to Upper	$\|\|(A)_{35-27}\| - \|(U)_{35-27}\|\| \rightarrow (A+1)_{8-0}$	.75
76	07	CDU	Characteristic Difference to Upper	$\|(A)_{35-27}\| - \|(U)_{35-27}\| \rightarrow (A+1)_{8-0}$	.75
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	4.250
76	13	DFD	Double Precision Floating Divide	(A,A+1) ÷ (U,U+1) → A,A+1	17.25 / 17.50
76	14	DFU	Double Load & Unpack Floating	Unpack (U,U+1): fixed-point part → A+1,A+2, exponent → A_{10-0}	1.50
76	15	DLCF	Double Load & Convert to Floating	Standardize and pack from fixed-point part in (U,U+1), exponent in $(A)_{10-0}$ and store in A+1,A+2	2.125
76	16	FEL	Floating Expand and Load	$(U)_{1,8,27} \rightarrow A,A+1_{1,11,60}$	1.00
76	17	FCL	Floating Compress and Load	$(U,U+1)_{1,11,60} \rightarrow A_{1,8,27}$	1.625
77	0-17		Illegal Code		–

+ *Times are given for alternate bank case only. In cases where instruction and operand are procured from same bank, add .75 usec. to execution time.*

* *For all comparison instructions, the first number represents the skip or jump condition, the second number is for no skip or no jump condition. For function codes 01-67, add .375 usec. to execution times for 6-bit and 12-bit writes.*

NOTE: Execution time for the Block Transfer and The Search instructions depends on the number of repetitions of the instruction required. The variance in all cases is .75K micro-seconds where K equals the number of repetitions; that is, K equals the number of words in the block being transferred or the number of words searched before a match is found.

APPENDIX C. DEFINITION OF INTERRUPTS

An interrupt is a special control signal generated to break the normal sequence of the execution of a set of instructions. Each interrupt is associated with a fixed, hardware defined core storage location connected with the event or circumstances causing the particular interrupt. The fixed address, to which program control is switched, provides an entry into a subroutine to process the particular interrupt.

INTERRUPT SEQUENCE AND PROGRAMMING INTERRUPTS

The occurrence of an interrupt causes the contents of the PSR (Processor State Register) to be stored into Control Regsiter 000_8 (XØ). All PSR designators will be cleared to zero except the BI, BD, and BS designators. The D_6 and D_7 designators will be set to one in the PSR. The storing and the modification of the PSR takes place during the interrupt sequence and before the execution of the instruction at the interrupt main storage location. The instruction executed as a result of the interrupt should have the i-designator set so that BI, BD, and BS of the worker program are ignored. After modification of the PSR, the contents of the particular interrupt location is executed without changing the contents of the P-Register unless the instruction in the interrupt location changes the contents of P, as is the normal case. Instructions which capture the contents of the P-Register and in addition transfer control to the entrance of a subroutine for processing the interrupt condition should be stored in the interrupt locations so program control can be returned to the proper place in the interrupted program after the interrupt is processed.

In the normal sequence of events (when an interrupt does not occur) the contents of the P-Register (when as modified by the PSR Register) is used by the control section of the 1108 to determine the main storage location from which to fetch the next instruction which is to be executed. A jump instruction causes the P-Register to be replaced entirely by the address from which the next instruction is to be taken. When an interrupt occurs, however, the location of the next instruction is determined not by the contents of the P-Register, but by the specific interrupt. Without changing the contents of the P-Register the next instruction is fetched from the interrupt location. Since normally the instruction is a jump instruction of the type which captures the contents of the P-Register as it jumps, the contents of the P-Register

as it was at the time the interrupt was honored will be captured. The contents of the P-Register will be then changed by the jump instruction to reflect the entrance of the subroutine which is to handle the interrupt. Thus it is necessary to provide this type of jump instruction at every interrupt location.

If programs are executed under control of the Executive Routine and a Worker Program is in control of the central processor, then the occurrence of an interrupt forces a change from the Worker to the Executive State of the system.

CLASSES OF INTERRUPTS

Interrupts are grouped according to the particular conditions causing interrupts to be generated. The three classes of interrupts are:

a) Input/Output (I/O) Interrupts

b) Fault Interrupts

c) Executive Return Interrupt

INTERRUPT LOGIC

Whenever an Input/Output or fault interrupt occurs or an Executive Return (72-11) instruction is executed, all I/O interrupts are disabled until an Allow All I/O Interrupts and Jump (74-07) instruction is executed. Fault interrupts are never disabled by any of the interrupts including the Executive Return (72-11) instruction. The Prevent All I/O Interrupts and Jump (72-13) instruction disables I/O interrupts only, fault interrupts and the Executive Return interrupt are not disabled.

INPUT/OUTPUT INTERRUPTS

DEFINITION OF I/O INTERRUPTS

An I/O interrupt is a signal sent by the I/O equipment to the UNIVAC 1108 Computer informing it of an abnormal condition or indicating the normal completion of an I/O operation, if it was requested (programmed with monitor) by the program that initiated the I/O operation. The interrupts caused by the Day Clock and Real-Time Clock indicate events that have occurred on the Clocks.

CLASSES OF I/O INTERRUPTS

I/O interrupts are subdivided into the following classes:

1. Monitor (Internal) Interrupts

    a. ISI Input Monitor Interrupt
    b. ESI Input Monitor Interrupt
    c. ISI Output Monitor Interrupt
    d. ESI Output Monitor Interrupt
    e. ESI Function Monitor Interrupt

Table C-1. Interrupt Address Assignment

CORE LOCATION -OCTAL-	INTERRUPT TYPE	CONTENTS OF P-REGISTER
210	Power Loss Interrupt	K+1
211	I/O ESI Control Word Parity Interrupt	K+1
212	I/O ISI Control Word Parity Interrupt	K+1
213	I/O Data Parity Interrupt	K+1
214-215	Not used	-
216	Day Clock Count	-
217	Day Clock Interrupt	K+1
220	ISI Input Monitor Interrupt	K+1
221	ISI Output Monitor Interrupt	K+1
222	ISI Function Monitor Interrupt	K+1
223	External Interrupt	K+1
224	ESI Input Monitor Interrupt	K+1
225	ESI Output Monitor Interrupt	K+1
226	Not used	-
227	ESI External Interrupt	K+1
230	External Interrupt Status Word	-
231	RTC Interrupt	K+1
232	External Synchronous Interrupt #1	K+1
233	External Synchronous Interrupt #2	K+1
234	Not used	-
235	Memory 2 Instruction Parity Fault Interrupt	K+1 or K+2*
236	Memory 3 Instruction Parity Fault Interrupt	K+1 or K+2*
237	Memory 4 Instruction Parity Fault Interrupt	K+1 or K+2*
240	ICR Instruction Parity Fault Interrupt	K+1 or K+2*
241	Illegal Instruction Interrupt	K+1
242	Executive Return Interrupt	K+1
243	Guard Mode Interrupt	K+1 or K+2
244	Not used	-
245	Floating Point Underflow Fault Interrupt	K+2
246	Floating Point Overflow Fault Interrupt	K+2
247	Divide Fault Interrupt	K+2

Last address minus-1 switch options determines memory 1 instruction parity Interrupt Location K+1 or K+2

An instruction at core location K was executed when the interrupt occurred.

* If Parity Errors occur during an I/O transfer, K+1.

2. External Interrupts

   a. ISI External Interrupt
   b. ESI External Interrupt
   c. Day-Clock Interrupt
   d. External Synchronous Interrupts 0 and 1
   e. Console Initiated External Interrupt

3. System I/O Interrupts

   a. Real-Time Clock Interrupt
   b. Power Loss Interrupt
   c. I/O Parity Error Interrupt

      1) I/O ESI Access Control Word Parity Interrupt
      2) I/O ISI Access Control Word Parity Interrupt
      3) I/O Data Parity Interrupt

I/O INTERRUPT PROGRAMMING

I/O interrupts become effective according to the Priority Schedule (Table C-2). The value of the P-Register captured when an I/O interrupt occurs will be K+1 (see Table C-1) where K is the address of the instruction executed when the interrupt occurred. The contents of the P-Register will be K if an instruction in the repeat mode is interrupted. Whenever an I/O interrupt occurs, all other I/O interrupts are disabled. That means they are generated, but they will not be honored until enabled by executing an Allow All Interrupts and Jump (74-07) instruction. Possible waiting I/O interrupts will then become effective according to the Priority Schedule (Table C-2). For details of Input/Output Sequences see the Input/Output section. In addition to the automatic disabling of I/O by the hardware of the central processor after any one of the I/O interrupts have occurred, the I/O interrupts can be program controlled. The instructions Prevent All I/O Interrupts and Jump (72-17) and Allow All I/O Interrupts and Jump (74-07) are used for controlling the occurrence of I/O interrupts. If a Prevent All I/O Interrupts and Jump instruction is executed when the central computer is operating in the Guard Mode (Worker State), I/O interrupts must be processed within 100 microseconds after they occurred. A Guard Mode fault interrupt will be generated if the I/O interrupts are not enabled within 100 microseconds after an I/O interrupt occurred.

INTERRUPT PRIORITY

Although all channels may be available for communications between the computer and the peripheral units at the same time, only one channel will actually be in use at any given instant. Since the peripheral device's speed of handling data is always lower than the one of the computer, it is possible for the computer to keep all or part of the peripheral equipment operating simultaneously. For this purpose, the computer operations are sequenced and synchronized by a priority control network within the Input/Output Section of the computer. Priority of access to the processor is resolved by a priority schedule (Table C-2). This handles situations in which two or more channels simultaneously attempt to communicate with the computer. When this occurs, access to the computer is granted on the basis of the following priority schedule:

Table C-2. Priority Schedule

1. Output Data Request (ODR)
2. Input Data Request (IDR)
3. Day Clock Increment
4. Input Data Request (Channel #15)
5. Real-Time Clock Decrement
6. Power Loss Interrupt
7. External Interrupt (ESI)
8. Input Monitor Interrupt (ESI)
9. Output Monitor Interrupt (ESI)
10. Real-Time Clock Interrupt
11. External Interrupt (ISI)
12. Day Clock Interrupt
13. External Interrupt on Channel #15 (ISI)
14. Input Monitor Interrupt (ISI)
15. Output Monitor Interrupt (ISI)
16. Function Monitor Interrupt (ISI)
17. External Synchronous Interrupt #0
18. External Synchronous Interrupt #1

When two or more channels simultaneously request the computer to process an operation of the same function priority level, access is granted to the lowest-numbered channel.

MONITOR INTERRUPTS

Monitor (internal) interrupts will be generated only if Input/Output instructions calling for this type of interrupts are programmed. I/O instructions calling for ISI and ESI input or output operations and ISI function transfers may be programmed "With" or "Without Monitor". Initiating I/O instructions "With Monitor" means that a Monitor (Internal) Interrupt will be generated upon completion of the data or function transfers, when initiated "Without Monitor" means that the Monitor (Internal) Interrupts will not be generated. The instructions calling for Monitor (Internal) Interrupts are:

Load Input Channel and Monitor (75-01)

Load Output Channel and Monitor (75-05)

Load Function On Channel and Monitor (75-11)

Since only one interrupt location is assigned for all I/O channels by the hardware for each Monitor Interrupt, but each type of Monitor Interrupt can occur on every I/O channel, a Store Channel Number (72-14) instruction must be executed after a Monitor Interrupt is received in order to determine the channel for which this interrupt is generated. Each I/O channel can be in either of two modes, the ISI and ESI Mode, which determines whether ISI or ESI Monitor Interrupts will be generated.

ISI INPUT MONITOR INTERRUPT

For ISI Input Monitor Interrupts one location for all I/O channels in the ISI mode is assigned, main storage location 220_8. This interrupt is generated when the Input Active/Inactive Flip-Flop for a channel is switched from the "Active" to the "Inactive" state. This switching is the result when the processor finds a word count W of the ISI Input Access Control Word to be equal to zero. The ISI Input Monitor Interrupt occurs only if the input operation was initiated by a Load Input Channel and Monitor (75-01) instruction.

ESI INPUT MONITOR INTERRUPT

For ESI Monitor Interrupts one location for all I/O channels in ESI Mode is assigned, main storage location 224_8. The condition for generating an ESI Input Monitor Interrupt is identical to the condition for generating an ISI Monitor Input Interrupt, except since normally more than one ESI Input Access Control Word is active on the ESI channels, the word count W of any one of the ESI Input Access Control Words turning to zero will cause the ESI Input Monitor Interrupt to occur.

ISI OUTPUT MONITOR INTERRUPT

For ISI Output Monitor Interrupts one location for all I/O channels in the ISI Mode is assigned, main storage location 221_8. This interrupt is generated when the Output Active/Inactive Flip-Flop for a channel is switched from the "Active" to the "Inactive" state. This switching occurs as a result of an Output Data Request (ODR) signal generated by an I/O subsystem and the central processor finding the word count W of the corresponding Output Access Control Word equal to zero. The ISI Output Monitor Interrupt occurs only if the output operation was initiated by a Load Output Channel and Monitor (75-05) instruction.

ESI OUTPUT MONITOR INTERRUPT

For ESI Output Monitor Interrupts one location for all I/O channels in the ESI Mode is assigned, main storage location 225_8. The condition for generating an ESI Output Monitor Interrupt is identical to the condition for generating an ISI Output Monitor Interrupt, except since normally more than one ESI Output Access Control Word is active on the ESI channels, the word count W of any one of the ESI Output Access Control Words equal to zero and an ODR signal generated on the corresponding channel will cause the ESI Output Monitor Interrupt to occur.

ISI FUNCTION MONITOR INTERRUPT

For ISI Function Monitor Interrupts one location for all I/O channels in the ISI Mode is assigned, main storage location 222_8. This interrupt is generated when the Function Mode Active/Inactive Flip-Flop for a channel is switched from the "Active" to the "Inactive" state. This switching occurs as a result of an ODR signal generated by an I/O subsystem and the central processor finding the word count W of the corresponding Output Access Control Word equal to zero. The ISI Function Monitor Interrupt occurs only if the Function Mode on that channel was initiated by a Load Function In Channel and Monitor (75-11) instruction.

EXTERNAL INTERRUPTS

External Interrupts will be generated under the following general conditions:

1. A subsystem receives a function word, but the I/O operation cannot be initiated.

2. An I/O operation is initiated within a subsystem, but it cannot be completed because of a condition which has arisen in the subsystem during the execution of the operation.

3. An I/O operation is completed in the subsystems but an error condition has occurred during the execution of the operation which the hardware of the subsystem recorded.

4. An I/O operation was completed satisfactorily, but the operation was initiated by a Function Word requiring an External Interrupt after normal completion of the I/O operation.

5. The Day-Clock when activated.

6. Signals sent via the External Synchronous Lines 0 and 1.

7. Manual intervention on the console.

Only one External Interrupt will occur in response to a Function Word sent to a subsystem, even if several conditions arise during the execution of an I/O operation that call for the generation of an External Interrupt. When an External Interrupt occurs on a subsystem, a Status Word will always be sent to the computer to specify the condition that caused the interrupt. The Status Word is stored automatically into main storage location 230_8, so that the status information is available in that main storage location when the External Interrupt occurs.

EXTERNAL INTERRUPT PROGRAMMING:

External Interrupts can be controlled by programming the Allow All Channel External Interrupts (75-14) and the Prevent All Channel External Interrupts (75-15) instructions. These instructions enable or disable all External Interrupts on all I/O channels. It is possible to get one more External Interrupt after executing the Prevent All Channel External Interrupts (75-15). This is because the instruction prevents all External Interrupts from coming into the central processor, however, an External Interrupt may already be pending in the central processor I/O hardware section. If the Prevent All Channel External Interrupts (75-15) or Allow All Channel External Interrupts (75-14) instructions are executed when the central processor is in the Guard Mode, a Guard Mode Fault Interrupt will be generated. After executing the Prevent All Channel External Interrupts (75-15) instruction the Allow All Channel External Interrupts (75-14) instruction must be executed to allow External Interrupts to occur again. (The Allow All I/O Interrupts and Jump (75-07) instruction does not enable External Interrupts after a Prevent All Channel External Interrupts (75-15) instruction was executed. Also, Executing the Prevent All I/O Interrupts and Jump instruction locks out all I/O Interrupts even though one may be pending. The Prevent All Channel External Interrupts (75-15) and the Allow All Channel External Interrupts (75-14) instructions disable or enable the Day-Clock External Interrupt, the External Synchronous Interrupts, and the External Interrupt generated when keying in a character on the console after the "Interrupt Enable" button on the 1108 console was depressed.

ISI EXTERNAL INTERRUPT:

For ISI External Interrupts one location for all I/O channels in the ISI Mode is assigned, main storage location 223_8.

ESI EXTERNAL INTERRUPT:

For ESI External Interrupts one location for all I/O channels in the ESI Mode is assigned, main storage location 227_8.

DAY-CLOCK EXTERNAL INTERRUPT:

Main storage location 217_8 is assigned for the Day-Clock External Interrupt. The Day-Clock can be turned on or off manually from the maintenance panel. If activated, the Day-Clock time is stored into main storage location 216_8 every 600 milliseconds after it was updated. In addition the Day-Clock External Interrupt is generated every 6 Seconds.

EXTERNAL SYNCHRONOUS INTERRUPT:

Main storage locations 232_8 and 233_8 are assigned to the External Synchronous Interrupts #0 and #1. When another computer or special external device sends a signal to the central processor via the External Synchronous Lines 0 or 1 the respective External Synchronous Interrupt will be generated. The External Synchronous Lines 0 and 1 are additional lines to the regular I/O channel data and signal lines. The Initiate Synchronous Interrupt (73-14) instruction with an A-Designator of 0 or 1 sends a signal on the Synchronous Lines 0 or 1 to the other computer.

CONSOLE INITIATED EXTERNAL INTERRUPT:

Keying in a character on the computer console after the interrupt enable button was depressed generates an External Interrupt on the channel to which the console is assigned (normally channel 15_{10}). This External Interrupt references main storage location 223_8 since the console normally is assigned to an ISI channel.

SYSTEMS I/O INTERRUPTS:

Systems I/O interrupts are I/O Interrupts but they cannot be classified as either Monitor or External Interrupts. They are

1. The Real-Time Clock Interrupts

2. The Power Loss Interrupt

REAL-TIME CLOCK INTERRUPT:

Main storage location 231_8 is assigned to the Real-Time Clock Interrupt. The clock can be enabled or disabled manually from the maintenance console. If activated, the Real-Time Clock value in Control Register 100_8 (lower half) is automatically decreased every 2^{-10} seconds (approximately equal to one millisecond). The Real-Time Clock generates an interrupt when the clock count value equal to zero is decreased by one.

POWER LOSS INTERRUPT:

Main storage location 210_8 is assigned to the Power Loss Interrupt. The Power Loss Interrupt is generated in the event of an electrical power loss in the computer. This interrupt allows for easy restarting of programs when power losses occur.

Upon receiving a Power Loss Interrupt, the computer will have approximately 40 milliseconds to get ready for the loss of electrical power. This time can be used to store all pertinent data from the Control Registers into main storage. When the "Power Down" occurs on the processor, the contents of main storage is not changed, but the contents of the Control Registers will be destroyed. When power is restored, execution of real-time programs and other programs can be resumed.

I/O PARITY ERROR INTERRUPT:

An I/O Parity Error Interrupt is generated if a parity error occurs when reading or writing ISI or ESI I/O Access Control Words. The 1108 contains a four-bit I/O parity register (I/O PR) that holds the number of the channel which has a Control Register or main storage parity error. The program can store this channel number by executing a Store Channel Number (72 J-14) instruction upon processing an interrupt at address:

211_8 I/O ESI Control Word Interrupt

212_8 I/O ISI Control Word Interrupt

213_8 I/O Data Parity Interrupt

Parity Error Interrupts which occur during I/O operations will be honored only if I/O interrupts are enabled. The 1108 I/O parity hardware can hold the error condition and channel number for one error; all following parity errors are locked out until an instruction is executed. The action taken for each type of parity error is listed following; if a parity error is already being held, steps 1, 2 and 3 will not be performed.

I/O ESI Access Control Word Parity Error:

1) Load Channel number into I/O PR.
2) Interrupt to address 211_8,
3) Lockout I/O interrupts,
4) Channel remains active.
5) The ESI Access Control Word in main storage is not up-dated; therefore, parity of that control word will normally be incorrect.
6) Subsystem is acknowledged; on output, one word of all 1's is sent to the subsystem and on input, the write into core memory is inhibited.

I/O ISI Access Control Word Error:

1) Load Channel number into I/O PR.
2) Interrupt to address 212_8,
3) Lockout I/O interrupts,
4) Subsystem is acknowledged; on output, a word of all 1's is sent to the subsystem and on input, the write into core memory is inhibited.
5) Deactivate channel and inhibit Monitor Interrupt.

I/O ESI and ISI Data Error:

1) Load channel number into I/O PR.
2) Interrupt to address 213_8.
3) Lockout I/O interrupts,
4) Subsystem is acknowledged. On output, the word in error is sent to the subsystem. Parity errors can occur on ESI input data because the ESI transfers are parity-checked, half-word transfers.
5) Channel remains active.

FAULT INTERRUPTS:

DEFINITION OF FAULT INTERRUPTS:

Fault Interrupts are signals generated to indicated machine faults or certain programming errors occurring in the central processor.

CLASSES OF FAULT INTERRUPTS:

Fault Interrupts are subdivided into the following classes:

1. Processor Error Fault Interrupts

   a. Instruction Parity Fault Interrupts

2. Program Error Fault Interrupts

   a. Illegal Instructions Fault Interrupt
   b. Guard Mode Fault Interrupt
   c. Floating Point Characteristic Underflow Fault Interrupt
   d. Floating Point Characteristic Overflow Fault Interrupt
   e. Divide Fault Interrupt

FAULT INTERRUPT PROGRAMMING:

Fault Interrupts can never be disabled by executing instructions or by the occurrence of any other type of interrupt. Whenever a Fault Interrupt signal is generated, the Fault Interrupt will be honored at the termination of the instruction being executed presently. Only one type of Fault Interrupt can be generated during the execution of an instruction. The occurrence of a Fault Interrupt disables all I/O Interrupts but other Fault Interrupts are not disabled.

MACHINE ERROR FAULT INTERRUPTS:

Parity is checked on data in both the main storage modules and the Control Registers. All main storage and Control Registers contain one parity bit per half word (18 bits). Upon detection of a main storage or Control Register parity error, the instruction in the corresponding Fault Interrupt main storage locations is executed. All Parity Error Fault Interrupts will sound the Systems Alarm and light the appropriate lights of the computer console.

INSTRUCTION PARITY FAULT INTERRUPTS:

Instruction Parity Fault Interrupt locations are assigned as follows:

Storage Module 1:	Last Address Switch options $0\text{-}7_8$ determine Fault Interrupt location.
Storage Module 2:	235_8
Storage Module 3:	236_8
Storage Module 4:	237_8
Control Registers:	240_8

The options for the Storage 1 Parity Error Fault Interrupt Location are determined by the Last Address Switch on the maintenance panel. This feature is needed because the 1108's can be expanded in 16K modules. Therefore, the last address will vary according to the size of the main storage. The Storage 1 Parity Error Fault Interrupt location will be the last address minus 1 of the machine. This address is specified by the three toggle switches on the maintenance panel.

The three toggle switches labelled 14, 15, and 16 represent bit location 14, 15, and 16. The "up" position represents a one. Therefore, the following are the addresses for Storage Module 1 Parity Error Fault Internal locations.

Last Address Switch Setting	Location of Storage Module 1 Error Fault Interrupts
0_8 = 000	Not Used
1_8 = 001	$77\ 776_8$ (used for 32K system)
2_8 = 010	$137\ 776_8$ (used for 49K system)
3_8 = 011	$177\ 776_8$ (used for 65K system)
4_8 = 100	$237\ 776_8$ (used for 81K system)
5_8 = 101	$277\ 776_8$ (used for 98K system)
6_8 = 110	$337\ 776_8$ (used for 114K system)
7_8 = 111	$377\ 776_8$ (used for 131K system)

Instruction Parity Error Fault Interrupts are generated as the result of an instruction, operand, or a control register read. A main storage or Control Register parity error will cause a Fault Interrupt as soon as possible according to the Event Flow of Instruction Execution. The instruction being processed is completed and written back into storage. The value in the P-Register which will be captured upon a Parity Error Fault Interrupt, will be as follows with respect to the instruction that caused the interrupt:

1. Control Register Parity Error Fault Interrupt:

   The contents of the P-Register that will normally be captured is K+1 or K+2. (P) = K+2 if the instruction was overlapped. X-Register incrementation is performed if specified and the arithmetic results are stored into Control Registers.

2. Instruction Read Parity Error Fault Interrupt:

   The contents of the P-Register captured will be K+1. The arithmetic results of the previous instruction will be stored into the Control Registers if the previous instruction was overlapped.

3. Operand Read Parity Error Fault Interrupt:

   The contents of the P-Register captured will be K+1 or K+2. (P) = K+1 if the instruction was not overlapped. (P) = K+2 if the instruction was overlapped. Parity errors can occur on all partial store operations. Partial stores of one-third and one-sixth words always generate new parity in the half word that is stored into. Half word stores generate parity bits for the half word involved only. Arithmetic results computed from the operands with incorrect parity are stored into control registers.

PROGRAM RECOVERY FOR INSTRUCTION PARITY ERROR FAULT INTERRUPTS:

When an Instruction Parity Error Fault Interrupt occurs, the operating program will have to be terminated. The only feasible method of recovery is to resume executing the interrupted program from a program established rerun point. The above mentioned values of the P-Register are useful only as a maintenance aid.

PROGRAM ERROR FAULT INTERRUPTS:

Upon detection of a Program Error Fault Interrupt, the instructions stored at the corresponding main storage locations will be executed. The value of the P-Register, which will be captured when this type of interrupt occurs, will be K+1 or K+2 depending on the type of the Fault Interrupt.

ILLEGAL INSTRUCTION FAULT INTERRUPT:

An attempt to execute an illegal instruction where the processor detects an undefined operation code, will cause an Illegal Instruction Fault Interrupt. Main storage location 241_8 is assigned to the Illegal Instruction Fault Interrupt. The value of the contents of the P-Register will be K+1 when this fault interrupt occurs. K is the address of the data causing this fault interrupt.

If an Illegal Instruction Fault Interrupt is generated and the processor is not in the Guard Mode, the System Fault Alarm and a light on the processor console will be turned on.

GUARD MODE FAULT INTERRUPT:

When the processor is operating in the Guard Mode, an attempt to execute any of the Executive Control Instructions and additional conditions listed below will cause an interrupt to main storage location 243_8.

1. Load Processor State Register (72-15)
2. Load Storage Limits Register (72-16)
3. Initiate Interprocessor (Synchronous) Interrupt (73-14)
4. Select Interrupt Location (73-16)
5. Load Channel Select Register (73-16)
6. All I/O instructions (75-00 through 75-13)
7. An attempted store into Control Registers 40_8 through 100_8 and 120_8 through 177_8. (With 1107 compatibility designator set, these CR's may be referenced without causing a Guard Mode Interrupt.)
8. A store attempt into write protected main storage locations and/or a load attempt from read protected main storage locations, depending on the realization of read-and-write or write-only modes of storage protection mode selected in the Processor State Register (PSR)
9. The disabling of I/O Interrupts for more than 100 microseconds.

In addition to the conditions given above, the Guard Mode Interrupt is caused by an attempted store into write protected main storage locations if the processor is not in the Guard Mode, but write protection was selected.

The value of the P-Register captured will be K+1 when this fault interrupt occurs.

FLOATING POINT CHARACTERISTIC UNDERFLOW FAULT INTERRUPT:

Whenever a characteristic underflow occurs during the execution of a floating point instruction, a Floating Point Characteristic Underflow Fault Interrupt will be generated. Main storage location 245_8 is assigned to this fault interrupt. The value of the P-Register captured will be K+2, where K is the address of the main storage location of the floating point instruction causing the interrupt. The Floating Point Characteristic Underflow Fault Interrupt is

dialed for double precision floating point instructions only by setting the D5-Designator in the Processor State Register (SR) to one. If D5 is set to one, no interrupt will occur and 72 binary zeros are stored into the result of the double precision floating point instruction.

FLOATING POINT CHARACTERISTIC OVERFLOW FAULT INTERRUPT:

Whenever a characteristic overflow occurs during the execution of a floating point instruction a Floating Point Characteristic Overflow Fault Interrupt will be generated. Main storage location 246_8 is assigned to this fault interrupt and the value of the P-Register captured will be K+2, where K is the address of the main storage of the floating point instruction causing the interrupt.

DIVIDE FAULT INTERRUPT:

A Divide Fault Interrupt may be caused by the execution of any of the fixed point divide instructions. A division by a positive or negative zero or any member causing a result greater than the registers used causes a Divide Fault Interrupt for fixed point and floating point instructions.

Main storage location 247_8 is assigned to the Divide Fault Interrupt. The value of the P-Register captured will be K+2, where K is the address of the main storage of the divide instruction causing the Divide Fault Interrupt.

EXECUTIVE RETURN INTERRUPT

A special instruction is provided for the worker program to turn control over to the Exec Program. Executing the Executive Return (72-11) instruction will cause the Executive Return Interrupt to be generated. Main storage location 242_8 is assigned to the Executive Return Interrupt.

UNIVAC

DIVISION OF SPERRY RAND CORPORATION

UP-4053