REFERENCE MANUAL

SLASH 6

DIGITAL COMPUTER

Original
September, 1976

COMMUNICATIONS AND
INFORMATION HANDLING

HARRIS CORPORATION Computer Systems Division
2101 Cypress Creek Road, Fort Lauderdale, Florida 33309 305/974-1700

LIST OF EFFECTIVE PAGES

TOTAL NUMBER OF PAGES IN THIS PUBLICATION IS: 273
CONSISTING OF THE FOLLOWING:

Page No.	Change No.	Page No.	Change No.	Page No.	Change No.
Title	Original				
A, B	Original				
i thru v	Original				
1-1 thru 1-17	Original				
2-1 thru 2-21	Original				
3-1	Original				
4-1 thru 4-32	Original				
5-1 thru 5-9	Original				
6-1 thru 6-8	Original				
7-1 thru 7-171	Original				
A-1 thru A-6	Original				

Insert Latest Revision Pages. Destroy Superseded Pages.

CAUTIONARY NOTICE

While the Manufacturer has attempted to detail in this manual all areas of possible danger to personnel in connection with the use of this equipment, personnel should use caution when installing, checking out, operating and servicing this equipment, especially when power is on. As with all electronic equipment, care should be taken to avoid electrical shock in all circuits where substantial currents or voltages may be present, either through design or short circuit. Caution should be observed also in lifting and hoisting equipment especially regarding large structures during installation.

The Manufacturer is specifically not liable for any damage or injury arising out of a worker's failure to follow the instructions contained in this manual, or his failure to exercise due care and caution in the installation, operation, checkout and service of this equipment.

CONTENTS

CONTENTS (Cont'd.)

CONTENTS (Cont'd.)

CONTENTS (Cont'd.)

ILLUSTRATIONS

TABLES

SECTION I
INTRODUCTION

1-1 SCOPE OF MANUAL

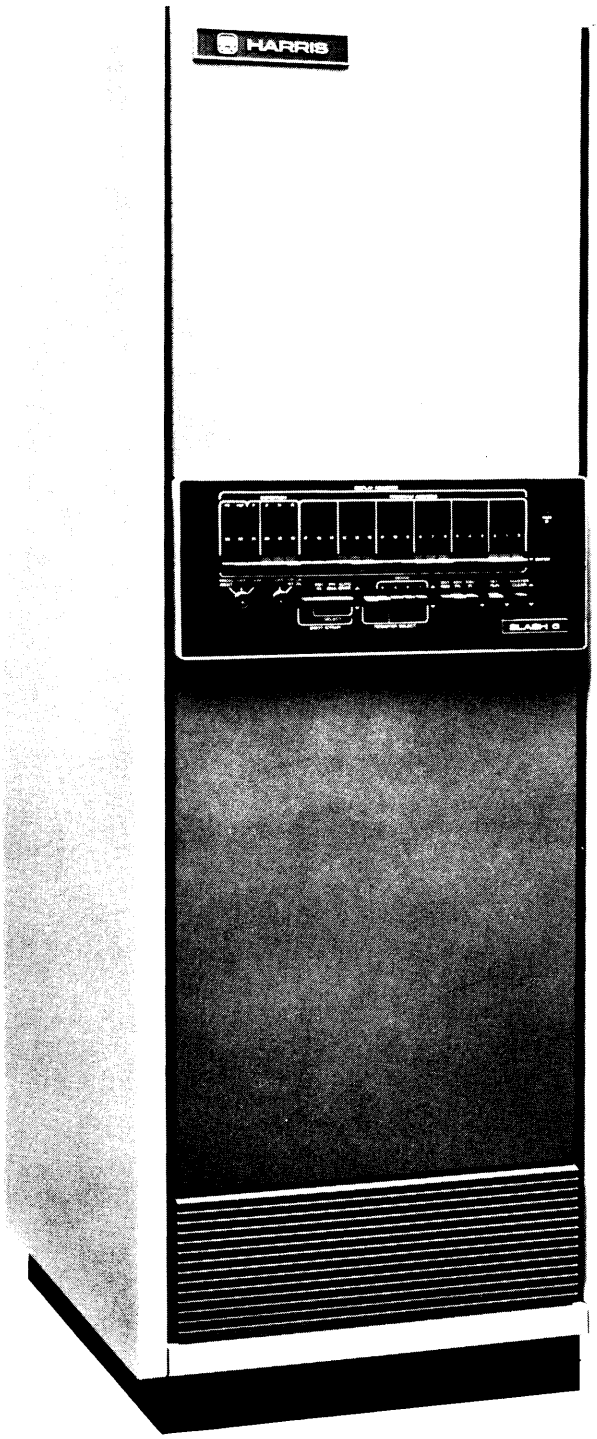
This manual contains reference material for the SLASH 6 digital computers designed and manufactured by Harris Corporation, Computer Systems Division, Fort Lauderdale, Florida. Included are descriptions of the computer's overall organization, central processing unit (CPU), memory configurations, priority interrupt system, input/output (I/O) channels, and instruction set. Various hardware options are also described; application and programming examples are provided where appropriate.

The material in this manual is oriented toward the user/programmer with a knowledge of computer fundamentals and terminology.

1-2 COMPUTER FEATURES

A SLASH 6 computer, shown in Figure 1-1, has the following operational features:

- Asynchronous CPU with instruction prefetch
- 24-bit word length
- Semiconductor Memory
- Memory parity error checking and correction code
- Hardware multiply, divide, and square root
- Boolean (single bit) processor
- Five general-purpose registers (three usable as index registers)
- Priority interrupt control system with up to eight executive traps and 24 external levels available
- Fully-buffered I/O channels for programmed data transfer and/or direct memory access
- Scientific Arithmetic Unit (SAU) - hardware for concurrent floating-point operation
- Power Failure protection for operating programs
- 120 Hz clock for system timing
- Optional Real Time 100 KHz clock.
- Optional Interval Timer



HT504

Figure 1-1. SLASH 6 Digital Computer

1-3 BASIC COMPUTER ORGANIZATION

1-3.1 Basic Operation

Figure 1-2 illustrates the functional relationship between major units of the SLASH 6 computer. The major functional units are: the central processing unit (CPU), memory, priority interrupt system, input/output (I/O) channels, control panel, and the optional Scientific Arithmetic Unit (SAU).

The SLASH 6 computer has a 24-bit fixed word length, a multiaccess bus structure, and an integral memory system. Operations are performed on, and from, 24-bit data and instruction words. In addition, the computer is capable of selective byte manipulation and performs Boolean functions on single, selected, bits. Two's complement arithmetic is performed on parallel, binary, fixed-point operands. Concurrent floating-point arithmetic is performed by the SAU.

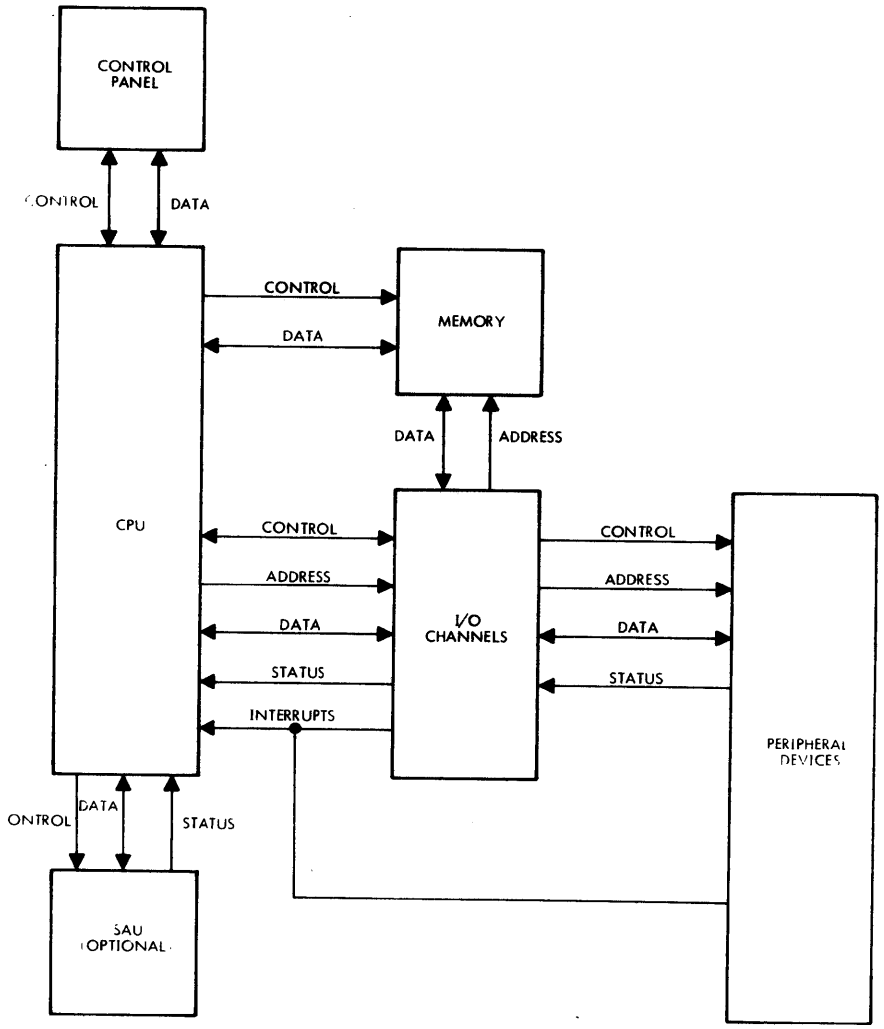
Data or instruction words may be retrieved from or stored in memory, retained in one of the CPU registers, or received from/transmitted to peripheral devices via the I/O channels. Prior to execution, instructions must be loaded into, and subsequently retrieved from, physical memory. Main memory is accessed on a double word boundary. This arrangement permits an instruction prefetch which reduces the effective access time of the memory system. In addition, the CPU employs an asynchronous cycle that automatically adjusts to the timing of the addressed memory module. If, for example, memory contention occurs, the CPU cycle will wait at a predetermined point until memory becomes available.

Memory may be accessed at the word, double-word, byte, and bit levels by the standard instruction set. The standard addressing technique divides memory into 32K - word sections. Up to 32K words per section may be directly addressed and up to 256K words can be accessed by indirect and indexed address references. Executable code is restricted to 65,536 (64K) words at any given time.

1-3.2 Central Processing Unit (CPU)

Included in the SLASH 6 CPU are several general- and special-purpose registers, an arithmetic section, timing and control logic, memory interface circuits, and I/O channel interface circuits. When the system includes an SAU, the CPU includes special circuits for CPU-SAU interface and communications.

Five general-purpose registers are included in a basic SLASH 6 CPU. These registers are employed in a variety of logical, arithmetic, and manipulative operations such as register-memory, memory-register, and register-register instructions. Three of the general-



BD1635-976

Figure 1-2. SLASH 6 Major Functional Units

purpose registers can be used for indexing in memory addressing functions. One register serves as the I/O communication register during single-word input/output operations. A double-word register is formed by combining two 24-bit registers; and a byte register is created by using the eight least-significant bits of one general-purpose register. When the optional Interval Timer is included in the CPU, the Timer (T) register becomes a sixth general-purpose register in various instructions.

Among the special-purpose registers are those associated with integral CPU functions such as addressing, instruction decoding, and temporary storage during data manipulation. Additional special-purpose registers are those supplied with: the bit (Boolean) processor option, Interval Timer option (T register, timing applications); Program Restrict and Instruction Trap option; and the Program Halt and Address Trap.

The arithmetic section consists primarily of a 24-bit parallel adder and several buses to permit data manipulation between the various registers and the adder. Arithmetic functions performed include addition, subtraction, multiplication, division, and square root computation. In addition, the adder output is employed in computing addresses during memory reference operations.

Instruction execution sequences are established and directed by the CPU's timing and control logic. This logic includes a crystal-controlled clock generator that provides precise timing for all instruction functions. Instruction words are retrieved from memory and retained in an instruction register for the duration of the operation. The control logic decodes these instruction words and provides the internal commands necessary for execution.

CPU memory interface circuits consist of address - and data-handling buses and registers, and parity generating/checking or error checking and correction code logic. Memory interface circuits include: a 48-bit data register that retains both the read and write data; an 18-bit address register to define the physical memory location to be accessed; data multiplexing logic to control read and write data handling; and address multiplexing and control logic for selecting the proper memory segment and a location within that segment. Data to be written (stored) in memory is applied via the system data bus. Data read (retrieved) from memory is applied to the CPU via the system data bus. Address inputs are applied to the memory interface via the system address bus. The address source may be the CPU memory address register, program address register (program counter), the adder or operand register in the arithmetic section, or one of the block transfer channels (UBC, XBC, IBC).

Communications between the CPU and the I/O channels are conducted via the channel interface logic in the CPU. This logic makes use of the system buses and one of the general-purpose registers in order to implement data and address flow between the CPU and I/O channels. Although an I/O channel conducts channel-unit communications independently and asynchronously, input/output operations such as channel-unit selection and activation, function commands, and status testing are initiated under program control.

When the Scientific Arithmetic Unit (SAU) is employed, CPU-SAU communications are conducted via interface circuits both in the CPU and SAU. Single-or double-precision transfers may be performed. All SAU instructions and data transfers are initiated on CPU timing. All concurrent floating-point arithmetic operations are performed on SAU timing.

The Power Fail Shutdown and Restart feature provides a means for saving operating programs in the event of a power failure and restoring original conditions when power levels return to normal. Two executive trap interrupt levels - one for "power down" and the other for "power up" - are supplied.

1-3.3 Memory Units

Storage of information, both instruction and data words, is a function of the memory modules. Semiconductor memory modules are used in the memory configuration and are available in increments of 16K-words. Total main memory capacity of the SLASH 6 is 256K-words. Refer to Section III for additional details concerning the SLASH 6 memory system.

1-3.4 Input/Output Operation

Input/Output (I/O) operations in the SLASH 6 consist of data, address, command, or status transfers between selected peripheral devices and the CPU or memory. All such operations are initiated under program control and are conducted, asynchronously, by an I/O channel. Various types of I/O channel cards may be installed in a SLASH 6 system. All channels in the system can be active simultaneously; and each channel may communicate with a maximum of 16 devices, one at a time.

An I/O operation is initiated by selecting and activating a channel, and one of its assigned peripheral devices, through the execution of a computer input/output instruction. (The SLASH 6 repertoire includes seven input/output instructions.) A specific I/O operation may involve: preparing a peripheral device for a subsequent communication; determining the device's operational status; or initiating a data transfer. Once activated, the I/O channel provides complete functional control over the operation.

Data may be transferred on a single-word basis (i.e., one data word per instruction) or automatically, in block of n words per operation. Block data transfers are performed by the Universal Block Controller (UBC), External Block Controller (XBC), or Integral Block Controller (IBC) channel. Each available type of I/O channel permits data transfer to (input) and from (output) the computer.

I/O operations may also be conducted on an interrupt basis through the use of interrupt logic in the channel(s). The channel interrupt system can be placed under program control and selectively enabled or disabled by an input/output instruction. Peripheral device functions may be connected directly to the computer's priority interrupt system, bypassing the channel interrupt logic.

1-3.5 Priority Interrupt System

The SLASH 6 contains an interrupt system that allows additional program control of input/output and internal CPU operations, and immediate recognition of special external conditions on the basis of priority. Receipt and recognition of an interrupt trigger permits normal program flow to be diverted to a subroutine that services the interrupt and returns the program to its normal sequence at the point where the interruption occurred.

Two interrupt groups - 0 and 1 - are available. Group 0 is reserved for internal CPU functions/options and is comprised of eight executive trap interrupt levels. Group 1 is reserved for external interrupts; the group may have up to 24 levels. A basic SLASH 6 computer is supplied with eight external interrupt levels. Sixteen additional external interrupts are available in increments of eight levels. All executive trap levels are associated with specific options (Paragraph 1-4) and are supplied when the option is installed in the computer.

1-3.6 100- or 120-Hertz Clock

When the system is operated under Disc Monitor System (DMS) software, the 100- or 120-Hz clock option is required. Continuously-generated interrupt triggers are placed under software control by enabling or disabling the associated external interrupt level. By this method, the clock may be used for various timing operations. The clock continuously transmits 100 or 120 interrupt trigger pulses per second (at rates of 100 Hz for system operating on 50 Hz power and 120 Hz for 60 Hz power).

1-3.7 PROM Bootstrap

Automatic program loading from a selected peripheral device is provided by the PROM Bootstrap feature. Through the use of console switches, the appropriate bootstrap program is loaded into memory. Once loaded, the bootstrap program will automatically load a minimum of one record from the appropriate device. Programs stored in the PROM provide for loading from disc, paper tape, magnetic tape, punched cards, magnetic tape cassettes (paper tape emulation), and flexible diskette.

1-3.8 Power Fail Shutdown and Restart

This feature provides a means for protecting operating programs in the event of a power failure and for restoring the original conditions when power levels return to normal.

1-3.9 Control Panels

There are two types of control panels available for use with the SLASH 6; the optional Programmer's Control Panel and the standard Turnkey Control Panel. The Programmer's Control Panel contains the facilities for manually starting and halting operations, entering data into memory and the various registers, and selecting registers for display and/or entry. Indicators on the panel provide display for the contents of registers and memory, system status, and other important functions. Designed primarily for OEM systems, the Turnkey Control Panel is offered for those system applications not requiring the full functional Programmer's Control Panel. Complete operating instructions for the control panels are contained in publication number 0840003.

1-4 COMPUTER OPTIONS

1-4.1 Installation of Options

A number of options are available for the SLASH 6 computer. Many of these options are designed as "plug-in" cards for field installation, if desired. The basic SLASH 6 is prewired to accept all such options. Unless otherwise indicated, additional details pertaining to CPU options may be found in Section II.

1-4.2 Input/Output Channels

Various types of I/O channels are available with the SLASH 6. Each channel is designed for a particular input/output data transfer application. A brief description of each type follows. A more detailed discussion of the I/O channels is provided in Section IV.

A. Programmed Input Output Channel (PIOC)

This is an I/O channel capable of implementing a single-word, eight-bit, parallel data transfer between the CPU and a suitable peripheral device. This channel has provisions for installing up to four unit interface controllers on the I/O circuit board. In addition, the PIOC can drive up to 12 additional remote device controllers. This board also contains a programmable Interrupt Generator which may be used in multi-processor installations. If required, one or two Real Time Clocks may be installed on the board.

B. Universal Block Controller (UBC)

A UBC channel implements and controls automatic data transfers between memory and a suitable peripheral device. The UBC contains two I/O ports with data transferred through each port in a 24-bit parallel word format. Each port provides either command chained block transfers or programmed I/O transfers. Chained block transfer capability permits the transfer direction to be reversed and a subsequent data block to be automatically transferred, without program intervention. Addressing and block size (number of words transferred) are established under program control. Once initiated, all UBC operations proceed automatically. When operating in the chained block mode, two word (48 bits) transfers, to and from memory, take place. Each port is also capable of programmed I/O operations in which single-word (24 bits) transfers take place between the CPU and peripheral device. The UBC can drive up to 16 external device controllers.

C. External Block Controller (XBC)

An XBC is similar in operation to one port of a UBC except that address control and block length are provided by an external device, and that no command chaining capability is provided. One XBC channel board can support up to eight external device controllers.

D. Integral Block Controller (IBC)

An IBC channel performs automatic data transfers in a manner similar to one port of a UBC. The IBC contains provisions for the installation of up to two interface controllers directly on the channel board. The IBC transfers data in a multiplex mode between the device controllers and memory. Only data chaining may be performed by the IBC.

1-4.3 Scientific Arithmetic Unit (SAU)

Available as an option with the SLASH 6, the SAU provides concurrent floating-point arithmetic capability independent from the CPU. A special repertoire of instructions is provided for CPU-SAU transfers and for performing double-precision, floating-point, computations.

The SAU contains its own registers for manipulating double-precision quantities and for selecting arithmetic status (condition) after the operation is completed. Data and condition information are displayed on the Programmer's Control Panel as a function of selectable shared indicators. An executive trap is provided with the SAU for detection of overflow/underflow conditions. Refer to Section VI for a more detailed description of the SAU.

1-4.4 Bit Processor

Capability is provided by the Bit Processor for selectively changing, testing, or performing logical operations on a single bit in memory.

1-4.5 Program Restrict and Instruction Trap

These two features are supplied as one integral option. The Program Restrict provision allows areas of memory to be selected under program control, protected from unauthorized access, and guarded against accidental modification. When the CPU is in a restricted environment, the Instruction Trap feature provides the means for preventing execution of a predefined group of instructions. Two registers, two executive trap interrupt levels, and the associated control logic comprise the option.

1-4.6 Interval Timer

The programmable Interval Timer option functions as an internal CPU timer that provides a method for regulating operating program segments and recording other intervals. Depending on the instruction used for its activation, the Internal Timer clocks either CPU time or real time. In addition to its timing applications, the Interval Timer provides the user with an additional 24-bit general purpose register that may be accessed through the standard instruction set.

1-4.7 Stall Alarm

Certain operations in the computer's instruction set and other internal conditions prohibit the recognition of external interrupts. A series of these instructions or conditions could, therefore, produce a situation where external interrupts are, in effect, "locked out". The Stall Alarm monitors all instructions and conditions in this interrupt-prohibiting category. If a series of these instructions or conditions has not been completed before the elapse of a predetermined time period, an executive trap interrupt is generated. The subsequent interrupt-processing routine may then examine the situation and take any necessary corrective action. The Stall Alarm option includes the appropriate control logic and is furnished with the associated executive trap interrupt.

1-4.8 Program Halt and Address Trap

This option provides for a program halt or an executive trap interrupt to occur at a specified address and under certain conditions. The address trap is used as an on-line debugging aid for use in applications such as breakpoint tracing. An address may be defined under program control so that when the address is referenced, an interrupt will be generated at the assigned executive level. The address trap may be enabled or disabled under program control.

1-4.9 Real Time Clock

This option provides the programmer with general purpose clock pulses that are independent of the mainframe clock pulses. With an accuracy of .05%, the real time clock pulses are available whether the CPU is in standby or not. The timing pulses can be used to measure user's program running time, or to generate periodic interrupts. Programming is accomplished through normal input/output commands. One or two real time clocks may be installed on the Programmed Input Output Channel (PIOC) board.

1-4.10 Priority Interrupt

The SLASH 6 priority interrupt system provides added control over I/O operations and other system functions. Recognition of special external conditions is performed on a basis of predetermined priority. The system consists of two separate interrupt groups - Groups 0 and 1. Group 0 includes eight executive traps that are assigned to, and supplied with, various hardware options. Group 1 can have up to 24 external interrupts. Eight are supplied as standard, and 16 are available as options. Complete details pertaining to the priority interrupt system are contained in Section V.

1-4.11 Computer Link

This option permits block data transfers between interconnected SLASH 6 computers in a dual computer installation. The computer link is particularly useful in real-time control applications involving dual computers. See Paragraph 4-7 for a more detailed description of the link option.

1-4.12 Multi-CPU Channel Adapter

Used in a multiple SLASH 6 configuration, the multi-CPU channel adapter allows peripheral devices to be shared by two or more computers. A detailed explanation of this option may be found in Paragraph 4-8.

1-4.13 Run Time Meter

The Run Time Meter records the time that power is applied to the SLASH 6, i.e., whenever the circuit breaker is in the ON position. Elapsed time is measured in hours and tenths up to 99,999.9 hours.

1-5 PERIPHERAL DEVICES

A full line of peripheral devices is available for use in SLASH 6 computer systems. Each device is equipped with a specially-designed interface controller for communication between the device and an appropriate input/output channel in the computer. The following types of devices are available.

- Teletypewriters
- Interactive CRTs (with Hard Copy options)
- Console Data Terminals (with Magnetic Cassette features)
- Paper Tape Readers
- Paper Tape Punches
- Card Readers
- Card Punches
- Line Printers
- Printer/Plotters
- Moving-Head Disc Storage Systems
- Cartridge Disc Storage Systems
- Fixed-Head Disc Storage Systems
- Disc Storage Modules
- Floppy Disc Storage Systems
- Magnetic Tape Systems (High -, Medium-, and Low-Speed)
- Synchronous and Asynchronous Communications Interfaces
- Remote Terminals

1-6 SOFTWARE

SLASH 6 software includes one of five operating systems (depending on the computer system configuration), seven languages, six support programs, a system utility package and, optionally, three remote job entry support packages to large, host computer systems. The SLASH 6 can also act as a host processor for any RJE terminal that supports IBM 2780 protocol.

A Resident Operating System (ROS) is supplied with a basic SLASH 6 computer. Disc and Tape Operating Systems (DOS and TOS) and a Disc Monitor System (DMS) are also available with the SLASH 6 system.

The Disc Monitor System can perform concurrent real-time, batch and interactive processing. DMS is expandable from a minimal batch configuration to a large-scale system that provides multiprogramming foreground/background processing, spooled I/O, local and remote interactive terminal support, dynamic memory allocation, timer-scheduled programs, dynamic file creation, inter-program communications, program segmentation and overlaying, reentrant foreground capability, automatic background checkpointing, program protection and system accounting functions.

Available languages, support programs, and the RJE packages are listed below.

Languages

- FORTRAN IV Compiler with extensions
- Interactive BASIC Compiler with extensions
- RPG II Compiler
- Harris MACRO Assembler
- SNOBOL 4 Interpreter
- FORGO (Diagnostic FORTRAN Compiler)

Support Programs

- Sort/Merge
- Indexed Sequential File Handler
- ACRONIM (Interactive Text Editor)
- Cross Reference
- DEBUG

- Trace
- Utilities

Remote Job Entry Packages

- CDC UT-200 for 6000/7000 series
- IBM 2780 for 360/370 series
- UNIVAC 1004 for 1100 series
- Host Capability for IBM 2780 protocol terminals

1-7 SUMMARY OF CHARACTERISTICS

The major operating characteristics and pertinent technical specifications of the SLASH 6 computer are summarized in Table 1-1.

Table 1-1. Characteristics and Specifications, SLASH 6 Computer

Characteristics	Specifications
Organization	Microprogrammed, general-purpose digital computer, single address, multiaccess central system bus structure, and buffered I/O channels.
CPU Word Length	24 bits
Arithmetic	Parallel, binary, two's complement, fixed-point number representation. Hardware multiply, divide, and square root. Concurrent floating-point hardware optionally available.
CPU Cycle Time	600 nanoseconds
Instruction Timing - Semiconductor Memory (microseconds)	Prefetched Words Operation / Register to Register / Memory Reference Register Transfers .300 .900 Add/Subtract .300 .900 Multiply 5.700 6.000 Divide 12.300 12.600 Algebraic Compare .600 1.200 Logical Compare .300 .900 I/O .300 N/A Shift .600+($\frac{N}{2}$X.300) N/A

Table 1-1. Characteristics and Specifications, SLASH 6 Computer (Cont'd.)

Characteristics	Specifications	
	Non-Prefetched Words	
Operation	Register to Register	Memory Reference
Register Transfers	.600	1.200
Add/Subtract	.600	1.200
Multiply	6.000	6.300
Divide	12.600	12.900
Algebraic Compare	.900	1.500
Logical Compare	.600	1.200
I/O	.600	N/A
Shift	$.600+(\frac{N}{2}X.300)$	N/A
Main Memory		
Semiconductor		
Type	MOS	
Minimum Size	16K	
Maximum Size	256K	
Increment	16K	
Word Length	48 bits (double word)	
Parity	Data parity Check or one-bit error correct	
Access	Random	
Power Fail	Battery back-up	
Addressing Modes	Direct to 32K. (Direct to 64K via long address instructions.) Indirect to 256K words (data only). Indexed (three registers) to 64K words. Byte indexed to 192K bytes. Immediate.	
Input/Output		
Programmed Data Transfer	Single word to/from CPU register; 8 or 24 bits.	
Automatic Data Transfer Operation	Direct memory access via UBC, XBC, or IBC.	

Table 1-1. Characteristics and Specifications, SLASH 6 Computer (Cont'd.)

Characteristics	Specifications	
Single Channel Transfer Rates (words/sec	Input	Output
UBC (1 port active)	1,626,000 (4.88MB/sec	1,300,000 (3.9MB/sec)
(2 ports active)	1,660,000 (5. MB/sec)	1,320,000 (3.96MB/sec)
XBC	444,444 (1.33MB/sec)	444,444 (1.33MB/sec)
IBC	666,666 (2MB/sec)	444,444 (1.33MB/sec)
I/O Command Modes:		
Normal	Normal operation for each channel type.	
Multiplex	Channel released to master/slave peripheral units. (Not available on XBC, or IBC).	
Offline	Channel I/O drivers turned off, allowing second CPU to share devices without need for peripheral switches. (Not available on IBC.)	
Reset	Resets Multiplex or Offline mode. Channel restored on line, unit selected. (Not available on IBC).	
Priority Interrupt System		
Internal interrupts	Maximum of eight executive traps, each supplied with associated option. Dedicated memory locations.	
External interrupts	Maximum of 24 levels, in eight-level increments; eight levels standard. All levels individually enabled/disabled, armed/disarmed, or triggered under program control. Dedicated memory locations.	
Interrupt response	1.8 microseconds (minimum) to 7.8 microseconds (maximum).	
Software interrupts	Interrupt generator allows external interrupt levels to be triggered under program control,	
Power Failure Protection	Power Fail Shutdown and Restart (option).	
Electrical Requirements		
Voltage (standard)	115/230 Vac ±10% or 120/208 Vac ±10%, 4 Wire, Single Phase	
(optional)	115 Vac ±10% or 120 Vac ±10%, 3 Wire, Single Phase	
Frequency	50 Hz or 60 Hz ±3 Hz	
Current (maximum)		
Basic System	10 A (115/230 or 120/208 Vac, 4 Wire)	
	13 A (115 or 120 Vac, 3 Wire)	
Full System	12 A (115/230 Vac or 120/208 Vac, 4 Wire)	
	22 A (115 Vac or 120 Vac, 3 Wire)	

Table 1-1. Characteristics and Specifications, SLASH 6 Computer (Cont'd.)

Characteristics	Specifications
Environmental Requirements	
Power	
Basic System	1.4 KW
Full System	2.5 KW
Operating Temperature	50°F to 113°F (10°C to 45°C), ambient air.
Storage Temperature	32°F to 122°F (0°C to 50°C), ambient air.
Operating Humidity	20% to 80%, relative (noncondensing).
Storage Humidity	20% to 90%, relative (noncondensing).
Operating Altitude	-1000 to +6000 ft. (-305 to +1829m).
Storage Altitude	-1000 to +15,000 ft. (-305 to +4572m).
Heat Dissipation (basic)	6150 BTU/hr. (1540 kg-cal/hr)
(maximum)	14650 BTU/hr. (3670 kg-cal/hr)
Cooling	Forced air provided by fans on each chassis.

SECTION II

CENTRAL PROCESSING UNIT

2-1 GENERAL DESCRIPTION

The SLASH 6 Central Processing Unit (CPU) is a single-address, 24-bit parallel word-oriented, stored-program processor. Operations performed by the CPU include data transfers, arithmetic computation, and logical manipulation. These operations are defined by instructions stored in, and retrieved from, physical memory. The specified operation is performed on single-word, double-word, byte, or single bit operands stored in memory or contained in one of the CPU registers. Data word formats, as defined by both hardware and software, are illustrated in Figure 2-1.

In addition to the general-and special-purpose registers, the CPU contains: an arithmetic section that performs the actual computation and logical manipulation of operands; and a Control section that retrieves and decodes instructions from memory and directs the functional processes of the system. The CPU also contains interface elements for communications with the other computer elements; e.g., memory, the I/O channels, the control panel, and the optional Scientific Arithmetic Unit (SAU).

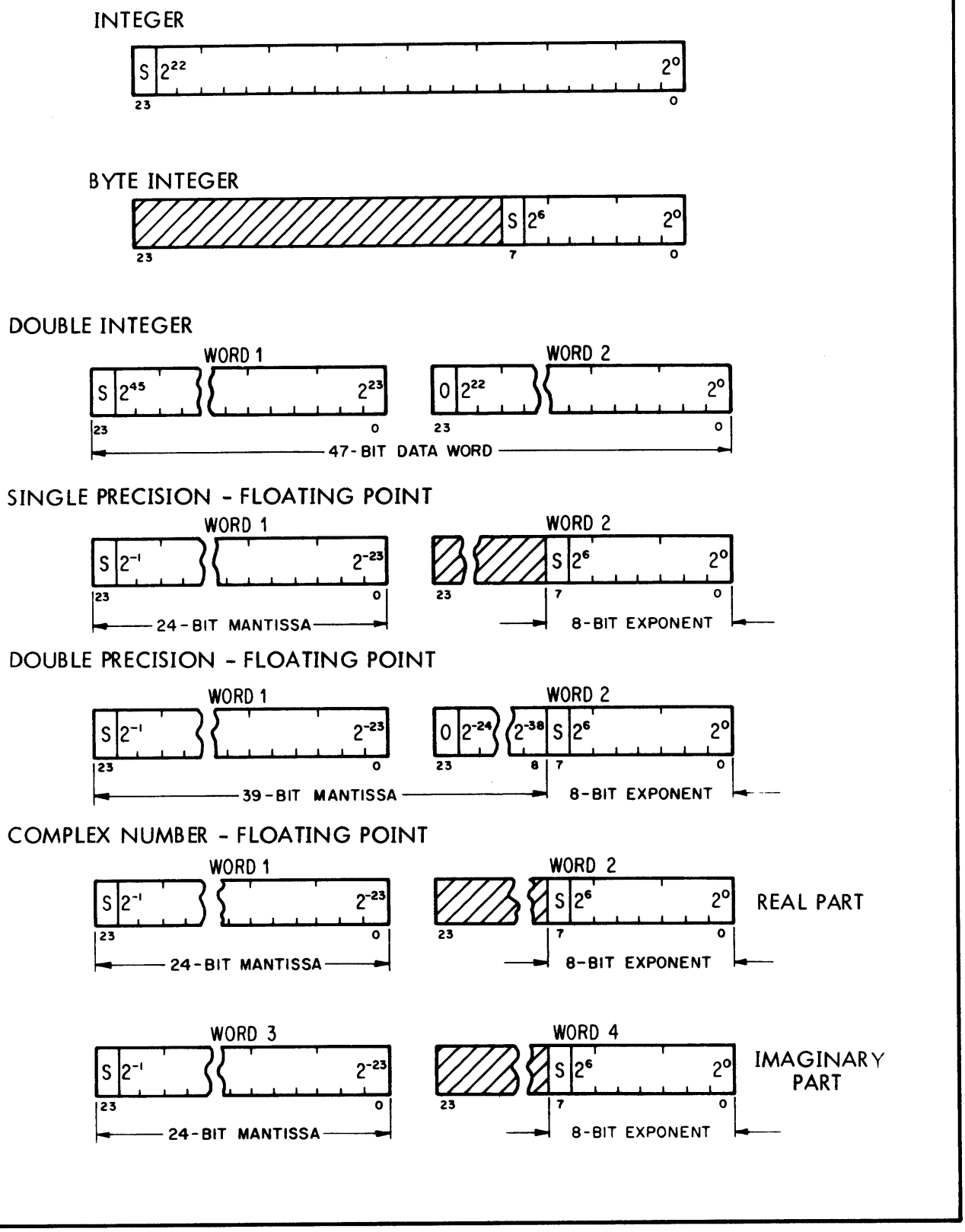
2-2 BASIC CPU REGISTERS

2-2.1 Introduction

The following paragraphs provide a brief description of the principle registers in a basic SLASH 6 CPU. Registers associated with Priority Interrupt system, SAU, and other options are described elsewhere, in the appropriate sections of this manual.

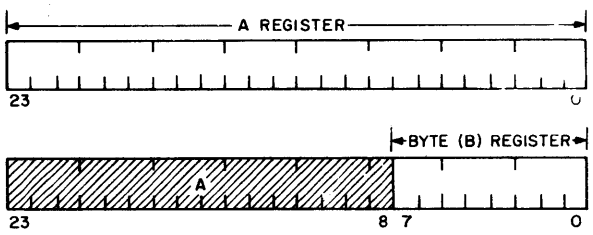
2-2.2 A and B Registers

Serving as the principal arithmetic accumulator, the 24-bit A register also functions as the Input/Output communication register during programmed (single-word) transfers between the CPU and peripheral devices. The A register has complete arithmetic and shift capability. Bits 7-0 of A form an 8-bit pseudoregister, termed the B (Byte) register. Both the A and B registers are accessible to the user by means of the instruction set and the Programmer's Control Panel.



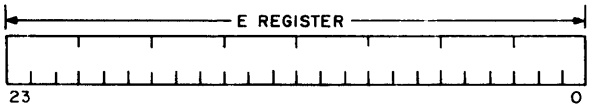
MI60-033-770B

Figure 2-1. Data Word Formats



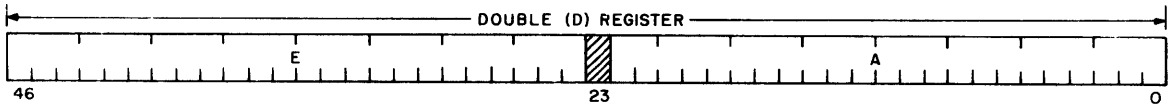
2-2.3 E Register

Employed as an extension of the A register for increased arithmetic and shift capability, the 24-bit E register also functions as a general-purpose storage element during various instructions. The E register is accessible through both the instruction set and the Programmer's Control Panel.



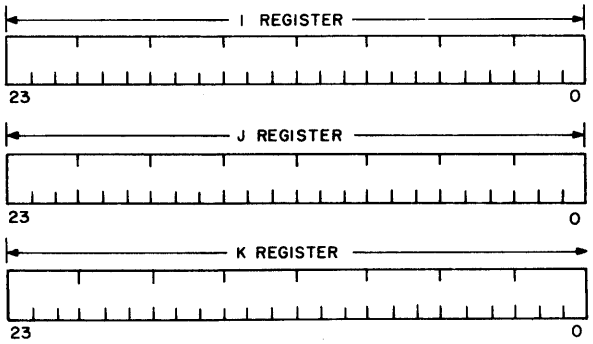
2-2.4 D Register

The D (Double) register is a 47-bit pseudoregister formed by combining A and E to provide double-precision arithmetic and shift capability. Registers A and E form the least- and most-significant halves, respectively, of the 47-bit double-precision quantity (bit 23 of A is not used). Several instructions provide direct access to the D register; Programmer's Control Panel entry, however, must be accomplished by a accessing the E and A registers in the proper format.



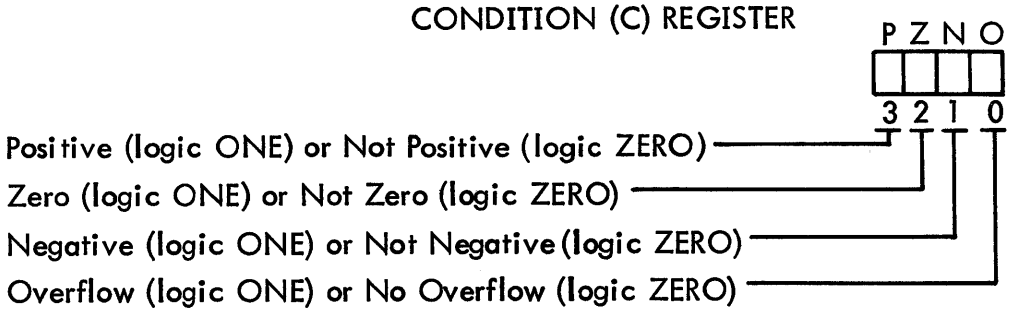
2-2.5 I, J, and K Registers

Each of these are independent, 24-bit general-purpose registers that can also be employed as index registers for address modification. The I, J, and K registers are directly accessible through the instruction set and the Programmer's Control Panel.



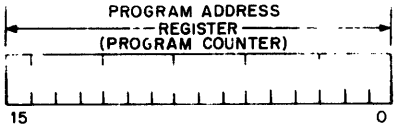
2-2.6 Condition Register

A 4-bit element that stores the results of specific operations, the Condition (C) register is accessible by means of several instructions. Display for the C register is provided by the Programmer's Control Panel.



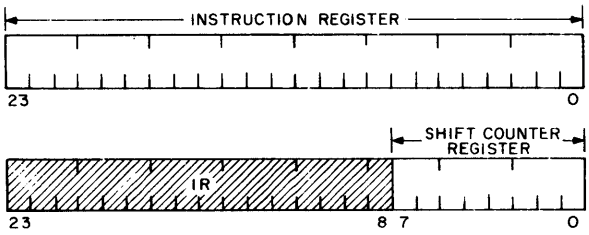
2-2.7 Program Address Register

Also called the Program Counter, the 16-bit Program Address (P) register retains the memory address from which the current instruction is fetched. A maximum of 65,536 memory locations can be accessed via the P register. Although not directly accessible through the instruction set, the contents of the P register can be modified through the execution of any of several Branch instructions. The Programmer's Control Panel provides direct entry and display for the P register.



2-2.8 Instruction Register and Shift Counter Register

Once an instruction has been fetched from memory, it is retained in the 24-bit Instruction register during decoding and execution. The Instruction register is not programmable. Bits 7-0 of the register serve as the Shift Counter register and is programmable via all shift instructions. Entry and display of the register is provided through the Programmer's Control Panel.



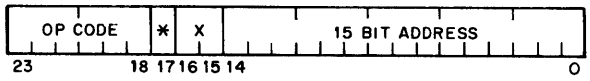
2-3 ADDRESSING FUNCTIONS

2-3.1 Basic Addressing Technique

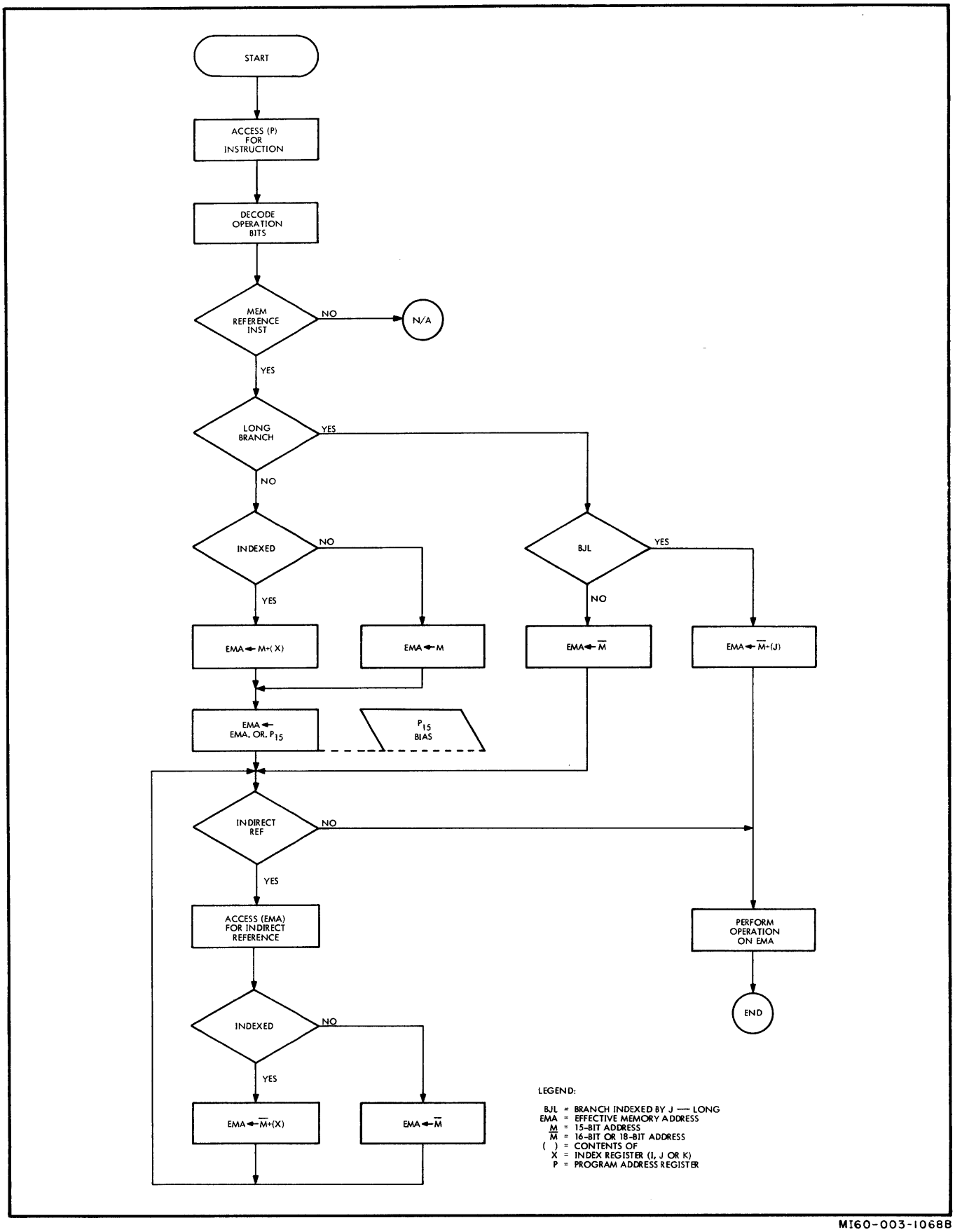
Total memory available to a SLASH 6 CPU is 262, 144 (256K) words. Because of the CPU's basic architecture and the corresponding addressing technique, executable code (programs) is confined to the lower 64K (0-65,536 words) of memory. However, memory above 65K may be addressed by means of special indirect references. Figure 2-2 illustrates the memory referencing sequence.

2-3.2 Direct Addressing

A standard memory reference instruction format is shown below. The 15-bit address field (bits 14-0) in the instruction word provides direct access to 32,768 (32K) words.



The addressing logic divides the lower 64K of memory into two areas: 0 - 32K; and 32K - 64K. Under this method, the most-significant bit (P_{15}) of the Program Counter is used to bias all direct address references. $P_{15} = 0$ specifies an address in the lower 32K, while $P_{15} = 1$ designates a location in the upper 32K of the 0 - 64K memory increment. By performing a logical-OR function between the immediate (direct) address reference and P_{15}, instructions may directly address up to 32K words within their respective sections of memory.



MI60-003-10688

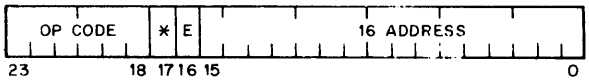
Figure 2-2. Memory Referencing Sequence

NOTE

An instruction in the last location of the lower memory section should not reference another address in the lower section. By the time the effective address is computed, the Program Counter will have advanced to bias the immediate address reference by 100000_8 to specify an effective address in the upper memory section.

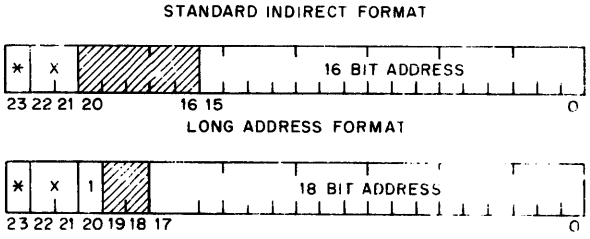
Modification of a 15-bit direct address by means of the indirect (*) and/or indexing (X) features can permit an instruction to address any memory location up to 256K words. These provisions are discussed in Paragraphs 2-3.3 and 2-3.4.

A special group of "Long Branch" instructions permits direct addressing up to 64K words. The instruction word format for this group is shown below. Note that these instructions may be modified by indirect references (*), but have no provision for indexing. Long Branch instructions are not biased by P_{15}. Bit 16 is used to extend the OP CODE.



2-3.3 Indirect Addressing

Indirect address references permit the CPU to access up to 256K words of memory. When a memory reference instruction is decoded, bit 17 (*) of the instruction word is examined. If bit 17 is set (ONE), an indirect address reference is indicated. An indirect reference signifies that the effective address (defined by the instruction word plus any index count) contains a second address rather than an operand. The word retrieved from memory when the effective address is cycled is treated as an indirect address word. Indirect address word formats are illustrated below.



The standard indirect format, with its 16-bit address field, permits access of up to 64K words. Up to 256K words can be accessed by the 18-bit field in the long address word. Neither type of indirect address is affected by the P_{15} address bias (Paragraph 2-3.2).

Bit 23 (*) of either indirect format may be set to specify another level of indirect addressing. Each level of indirect reference may be individually indexed to provide further address modification.

2-3.4 Indexing

A direct or indirect address reference may be modified by indexing. This operation adds the address in the current instruction or indirect reference to the contents of a specified index register (I, J, or K) to determine an effective address. A two-bit field (X) in the instruction or indirect reference specifies which register will be employed in each indexing operation. Figure 2-3 provides some examples of indexed addressing.

In the lower 32K memory section ($P_{15} = 0$), immediate address references may be indexed to access up to 65, 536 words. However, instructions in the 32K - 64K section of memory ($P_{15} = 1$) may not reference the lower section by indexing since all immediate address references will be biased by 100000_8.

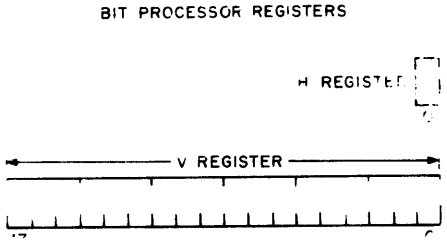
2-4 BIT PROCESSOR

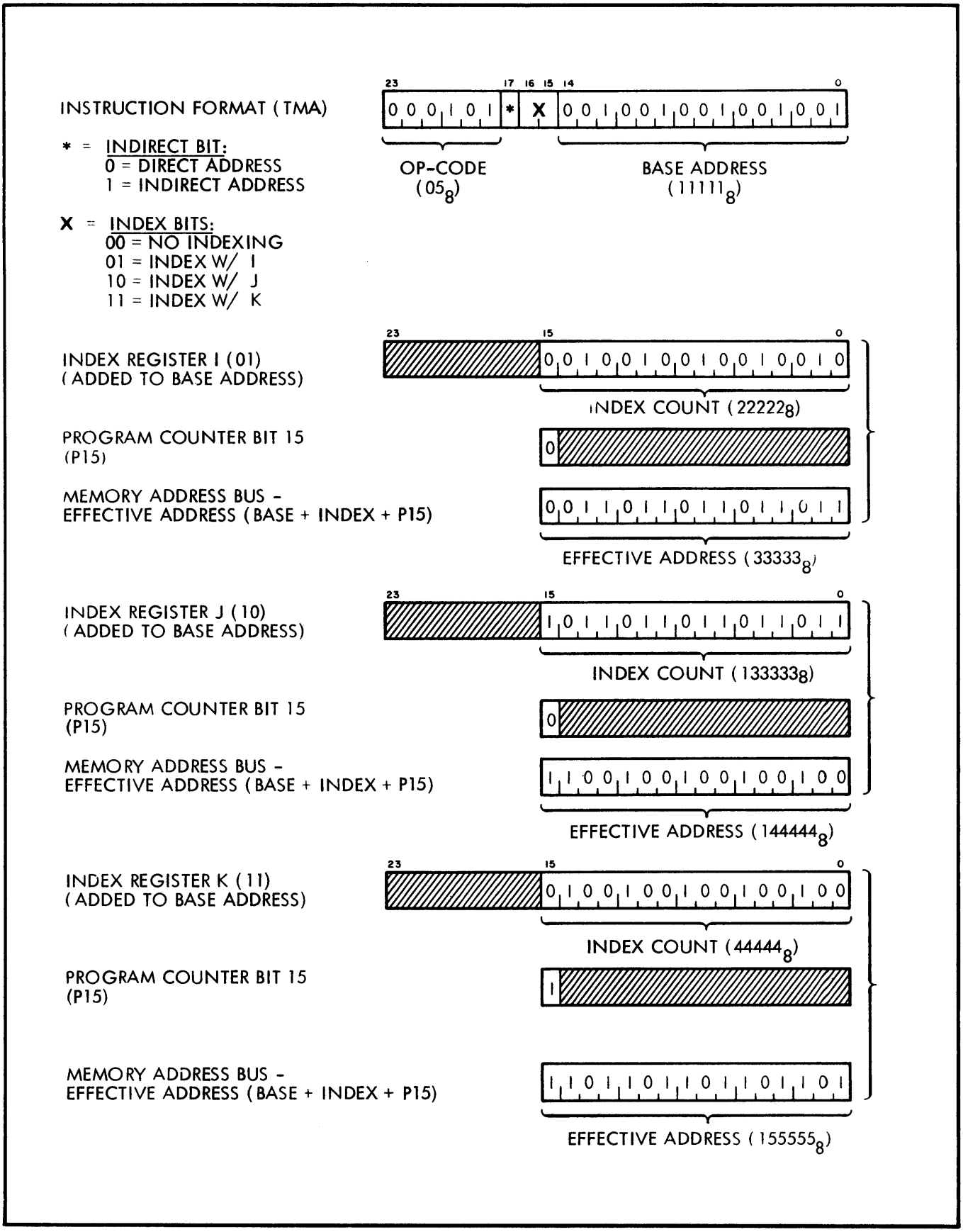
2-4.1 General Description

The bit processor option consists of the single-bit H register, an 18-bit V register (base register), and the associated control logic. The bit processor provides the capability to selectively change, store, or test a bit from memory.

2-4.2 Bit Processor Registers

Two registers are associated with the Bit Processor feature. A single-bit element, the H register, retains the bit selected for use in the operation. The 18-bit V register is employed to store a base address that is, in turn, used to define a memory location from which the designated bit will be retrieved. Both the H and V registers are directly programmable via the special group of Bit Processor instructions. Provision is made on the Programmer's Control Panel for entry and display of the Bit Processor registers. The registers are entered and displayed simultaneously, with the V register contents displayed in bit positions 17-0 of the display register indicators and the H register contents displayed in bit position 18.





MI60-028-9768

Figure 2-3. Examples of Indexed Addressing

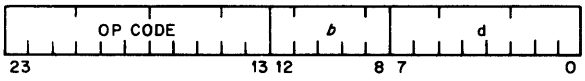
2-4.3 Operational Description

The 18-bit V Registers is loaded with a base address which specifies a memory location to be manipulated. This is accomplished by transferring an 18-bit memory address from the K register. The instruction word further defines the memory location, the specific bit, and the operation to be performed.

After the operation is performed on the selected bit, the results are displayed in the Condition register. Refer to Section VII, Paragraph 7-13.

2-4.4 Program Control

Two types of instructions are associated with bit processor operations. The first (shown below) specifies a displacement (bits 7-0) to be added to the base address (V Register contents) to specify the location to be accessed. Bits 12-8 (binarily coded) are used to select a specific bit to be used in the operation. The OP CODE is defined in bits 23-13.



The second word format is used for bit movement or transfers where a specific bit from memory is not required. Bits 23-12 contain the OP CODE; the remaining bits are undefined.

2-4.5 Bit Processor Instruction Set

The bit processor (Boolean function) group of instructions include branches, logical manipulation, and interrogation of a specified bit selected from an effective memory address or the H register. The following instructions are included in the bit processor group.

MNEMONIC	INSTRUCTION
DMH	Dot Memory with H
DNH	Dot Not (memory) with H
FBM	Flag Bit of Memory
NHH	Negate of H to H
OMH	Or Memory with H

MNEMONIC	INSTRUCTION
ONH	Or Not (Memory) with H
QBH	Query bit of H
QBM	Query bit of Memory
TFH	Transfer Flag to H
THM	Transfer H to Memory
TKV	Transfer K to V
TMH	Transfer Memory to H
TVK	Transfer V to K
TZH	Transfer Zero to H
XMH	eXclusive-or Memory with H
XNH	eXclusive-or Not (memory) with H
ZBM	Zero Bit of Memory

2-5 PROGRAM RESTRICT AND INSTRUCTION TRAP (OPTION)

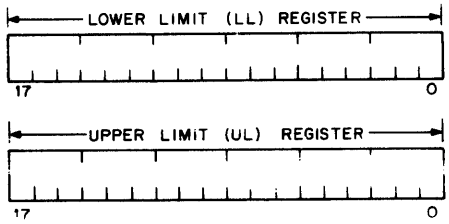
2-5.1 General Description

The Program Restrict and Instruction Trap option allows areas of memory to be selected under program control and protected from unauthorized access. The instruction trap provides a means for preventing the execution of a predetermined group of instructions. Two registers (18 Bit), two executive trap interrupt trigger circuits, and the associated control logic comprise this option. Control is maintained by two instructions: Transfer Double register to Limit register (TDL) and Transfer Limit register to Double register (TLD).

2-5.2 Program Restrict Registers

Two 18-bit registers are provided with the Program Restrict and Instruction Trap option. These registers define the upper (UL register) and lower (LL register) limits of a memory area that is restricted from unauthorized access. Both registers are programmable and may be entered and displayed via the Programmer's Control Panel.

PROGRAM RESTRICT REGISTERS



2-5.3 Operational Description

The CPU operates with the program restrict system enabled or disabled depending on the position of the PROG REST/OFF/VM key switch. When the restrict system is disabled, all memory is accessible.

When the restrict system is enabled, the computer operates in three distinct modes as established under program control. The three program restrict modes are defined below.

a. Unrestricted (Mode 0) - Programs working in this mode may access and alter any location in memory.

b. Restricted/Privileged (Mode 1) - Programs operating in this mode may access and load from any location in memory; however, Mode 1 programs may not alter the contents of, or transfer control to, any memory location outside the specified limits.

c. Restricted/Unprivileged (Mode 2) - Programs operating in this mode may not reference, in any manner, any memory location outside the specified limits.

Once established, the restrict mode is maintained by two flags operating concurrently. The mode flags can be set to one-of-three significant states to establish Mode 0, 1, or 2.

Control over the restrict system is maintained by the program restrict flag (PRF). The PRF operates under the condition and in the manner outlined below:

a. The PRF may be set only when the restrict system is enabled (i.e., when the PROG REST/OFF/VM key switch is in the PROG REST position).

b. When in Mode 1 or 2, the PRF is set when an instruction transfers control into the restricted area of memory.

c. When MASTER CLEAR is depressed, the PRF is set and Mode 2 is established automatically. MASTER CLEAR also clears the limit registers. Once a violation is made, the only way of recovering manually is by placing the PROG REST/OFF/VM key switch to the OFF position and then depressing the MASTER CLEAR switch.

d. Priority interrupts reset (turn off) the PRF, rendering the mode flags ineffective until control is returned to the restricted area and the PRF is set again.

e. When the PRF is set, executive trap interrupt 2 (Group 0, Level 2) occurs if a restrict violation takes place, except when operating in the Halt mode. (A restrict violation consists of any attempt to violate the conditions established

by the modes.) The one exception to this is the Branch and Link Unrestricted (BLU) instruction. The BLU instruction has been implemented as an executive call to allow restricted programs to communicate directly with the resident operating system without being trapped. The BLU instruction resets the PRF and transfers control unconditionally to the address specified by the instruction.

f. When the restrict system is enabled and operating in Mode 1 or 2, the console HLT/RUN switch can be activated only if the PRF is set. (This prohibits halting the machine during an interrupt routine.) If the CPU is operating in Mode 0, the only method of halting operation is to disable the restrict system (turn the key switch off) and activate the HLT/RUN switch.

g. If the CPU is being controlled manually (Halt mode) and the restrict switch is in Mode 1 or 2 (PRF set), violations will be treated as follows:

1. If the violating instruction is a branch or an attempt to transfer an operand into memory, it will be treated as a No OPeration (NOP) instruction.

2. If the violation attempts to transfer an operand from memory, all ZEROs will be retrieved.

3. If the normal advance of the Program Address causes the violation, all ZEROs will be retrieved and the Program Counter will be advanced.

4. If an attempt is made to enter an address into the Program Address Register that is outside the bounds specified by the Limit Registers, the Program Address Register will be loaded, and ZEROs will be retrieved from memory.

2-5.4 Program Control

The restricted area of memory is defined by two special registers, the Lower Limit (LL) and Upper limit (UL) registers. Each register retains an 18-bit address that defines one limit of the restricted area.

Two instructions, Transfer Double to Limit registers (TDL) and Transfer Limit Registers to Double (TLD), are provided for operating the limit registers. The limits are defined by executing a TDL instruction where D = E and A; bits E17-E0 specify the lower limit and A17-A0 specify the upper limit. The TDL instruction also establishes the restrict mode by setting the mode flags with Bits A_{21} and A_{22}. The bit configuration determines which mode will be established as shown on the following page.

A_{22}	A_{21}	Mode
0	0	0
0	1	1
1	0	2
1	1	0

NOTE

If an attempt is made to execute the TDL instruction while in Mode 1 or 2, the instruction trap is activated.

The TLD instruction provides a method of saving the contents of the limit registers plus the status of the mode flags. The contents of the LL register are transferred to bits E17-E0 and the contents of the UL register are transferred to $A_{17} - A_0$. The mode flag bit configuration is retained in bits A_{22} and A_{21}.

2-5.5 Instruction Trap

The instruction trap is enabled and disabled by the PROG REST/OFF/VM key switch. When the PRF is off, the instruction trap is inhibited.

If an attempt is made to execute any of the instructions listed below with the PRF on, an interrupt occurs at Group 0, Level 3. The interrupt routine may then examine the trapped instruction and determine what action is to be taken. The affected instructions are:

a. Halt (HLT)
b. Output Data Word (ODW)
c. Input Data Word (IDW)
d. Output Command Word (OCW)
e. Input Status Word (ISW)
f. Output Address Word (OAW)
g. Input Address Word (IAW)
h. Input Parameter Word (IPW)
i. Hold eXternal Interrupts (HXI)
j. Release eXternal Interrupts (RXI)
k. Unitarily Arm group 1 interrupts (UA1)
l. Unitarily Disarm group 1 interrupts (UD1)
m. Unitarily Enable group 1 interrupts (UE1)
n. Unitarily Inhibit group 1 interrupts (UI1)

o. Transfer Double to group 1 (TD1)
p. Transfer Double to group 1 (TD4)
q. Transfer Double to Limit Registers (TDL)

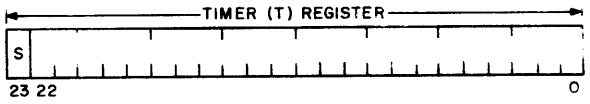
2-6 INTERVAL TIMER (OPTION)

2-6.1 General Description

The programmable Interval Timer option consists of a 24-bit register (T Register), a clock, and associated control logic. The timer can be preset and subsequently released, under program control, to measure elapsed processor (CPU) time or clock (real) time.

2-6.2 Timer Register

Supplied with the Interval Timer option, the 24-bit Timer (T) register operates as a counter in two distinct modes of operation. When not used for timing functions, the T register functions as an additional general-purpose register that can be accessed through the instruction set. Entry and display for the T register is provided via the Programmer's Control Panel.



2-6.3 Operational Description

A self-contained clock generates the 1 microsecond pulses used to strobe the timer. In either mode of operation, a positive count is loaded into the T register and decremented once for each elasped period of 1 microsecond. When the count reaches zero, an Executive Trap Interrupt is generated at Group 0, Level 5. A maximum count of $8,388,608_{10}$ (40000000_8) may be loaded into the register. With a resolution of 1 microsecond per count, a maximum time interval of 8.388608 seconds is available.

2-6.4 Program Control

Interval Timer operation is controlled by three instructions: Hold Interval Timer (HIT); Release Processor Time (RPT); or Release Clock Time (RCT). A HIT instruction will prohibit the start of any timing sequence or halt any in-process timing operation until the timer is released by an RPT or RCT instruction. The RPT instruction releases the timer for measuring elapsed

processor (CPU) time. In this mode, counting is inhibited during block controller channel memory cycle stealing operations or whenever any interrupt is active. Clock (real) time operation, where the timer counts continuously regardless of CPU condition, is initiated by an RCT instruction.

2-7 STALL ALARM (OPTION)

The Stall Alarm option is enabled and disabled by the OFF/CP.LK. ST.AL. key switch on the control panel. When the Stall Alarm is disabled, normal CPU operations take place. Once the Stall Alarm is enabled, a 128-cycle counter is activated whenever certain instructions are executed or certain operating conditions are encountered. The counter is incremented once each CPU cycle until the specified instruction(s) or conditions are removed. If the instructions/conditions are still present after 128 machine cycles, an executive trap interrupt is generated at Level 4 of Group 0.

The following instructions and/or CPU conditions will activate the Stall Alarm counter:

1. Unitarily Arm group 1 interrupts (UA1)
2. Unitarily Disarm group 1 interrupts (UD1)
3. Unitarily Enable group 1 interrupts (UE1)
4. Unitarily Inhibit group 1 interrupts (UI1)
5. Transfer Double to group 1 interrupts (TD1)
6. Transfer Double to group 1 interrupts (TD4)
7. Update Stack Pointer (USP)
8. Branch and Save return - Long (BSL)
9. Hold eXternal Interrupts (HXI)
10. Hold interrupts and Transfer I to memory (HTI)
11. Hold interrupts and Transfer J to memory (HTJ)
12. Hold interrupts and Transfer K to memory (HTK)
13. EXecute Memory (EXM)
14. Release eXternal Interrupts (RXI)
15. Transfer Registers to Memory (TRM)
16. Transfer Memory to Registers (TMR)
17. A halt condition
18. An indirect memory cycle.

Each of the preceding instructions prohibits the recognition of external interrupts for a period of one cycle following completion of the instruction. Executing a series of these instructions sequentially will lock out external interrupts for the entire series. Multilevel indirect addressing can produce a similar effect, since the instruction must satisfy all address references before completion. A halt condition - whether as a result of a programmed halt, operator action, or memory parity error - also prohibits external interrupt recognition by the CPU.

If a power failure occurs, the Stall Alarm becomes disabled. However, when power is restored, the Stall Alarm is re-enabled and operations continue in a normal routine.

With the exception of an EXM instruction or an indirect cycle, the monitored operation is allowed to complete its sequence before the executive trap assumes control. An EXM chain (where an EXM instruction references another EXM which, in turn, specifies a third etc.) has the same overall effect as an indirect chain in that all references must be completed before the sequence is complete. Therefore, if an EXM or indirect cycle is in process when the executive trap is generated, the Stall Alarm logic automatically terminates the sequence. If a block controller channel is transferring data into memory when the executive trap interrupt is generated, the current cycle is completed before termination occurs and the trap takes control. If a halt condition is in effect when the executive trap interrupt is generated, the Stall Alarm logic automatically forces the CPU into a run mode.

2-8 100 OR 120-HERTZ CLOCK (STANDARD)

This clock continuously transmits 120 or 100 mainframe interrupt signals per second, depending on power line frequency. The interrupt signal is controlled completely by enabling (or disabling) the assigned CPU interrupt level. The first interrupt following an enable signal will occur in less than 1/120 (1/100) of a second because the clock never stops transmitting signals; however, all subsequent interrupts will be precisely 1/120 (1/100) seconds apart.

The accuracy in using this clock is a function of the user interrupt routine logic. For example, if the clock is used to update a "time-in-seconds" counter by adding one count every 120 (100) interrupts, the "current time" at any given query will be accurate within 1 second. If, however, the counter is updated each interrupt - and divided by 120 (100) when "current time" is queried, the accuracy will be within 1/120 (1/100) of 1 second.

A simple example of coding, where the clock is assigned to priority interrupt Group 1, Level 22, is as follows:

```
*.....                          . Initialize Clock Routine
INITCT      TMA = B22           . (A) = Bit 22
            TME = B22           . (E) = Bit 22
            UA1                 . Arm L22/G1
            UE1                 . Enable L22/G1
            TZM CLOCK T         . Zero Clock Time
            BUC 0, J

*.....                          . Interrupt Routine
CLOCK IR    ***                 . Enter
            AUM CLOCK T         . Increment Clock Time
            BRL* CLOCK IR       . Restore C register and Exit

*.....                          . Current Time Routine
CTIME       TMA CLOCK T
            ESA
            DVO 120
            BUC 0, J            . Return: (A) = Seconds
                                          (E) = Remainder
```

2-9 PROM BOOTSTRAP (STANDARD)

The PROM Bootstrap automatically stores in memory a loader program that permits a more complex program to be stored. Any program can be loaded as long as it is in bootstrap format; however, the most common application is to load a loader program which allows other programs, operating systems, diagnostics, or other data to be stored in selected memory locations. Eight sources (bootstrap options) are selectable for transferring a program to memory via a selected peripheral device: paper tape; cassette (paper tape emulation); disc; card reader (word mode); card reader (block mode); magnetic tape; and flexible diskette. The operation of the bootstrap is implemented at the control panel. The specific device may be selected with the BOOTSTRAP SELECT switches and stored in memory by depressing the BOOTSTRAP ENA switch. A description of the bootstrap operation and individual bootstrap programs are documented in the SLASH 6 Operator's Manual, 0840003.

2-10 POWER FAIL SHUTDOWN AND RESTART (STANDARD)

This feature saves the operating program in the event of a power failure and provides program restoration and restart when power levels return to normal. This feature is applicable to semiconductor memories with battery back-up.

The shutdown circuits monitor the input AC power source for amplitude fluctuations. A decrease in AC voltage below the specified level causes an executive trap interrupt to be generated at Group 0, Level 0. If semiconductor memory with battery back-up is installed in the computer, the memory will be switched to battery. One millisecond after the interrupt, a Master Clear signal is generated to complete the shutdown process. In the one-millisecond interval between interrupt and final shutdown, the interrupt-processing routine must save the operating program along with parameters for returning to the point of interrupt.

When the AC power level returns to its nominal level, a restart signal is generated to begin the restore process. The restart signal generates an executive trap interrupt at Group 0, Level 1.

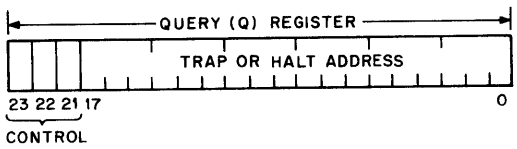
2-11 PROGRAM HALT AND ADDRESS TRAP (OPTION)

2-11.1 General Description

This option provides address trap or program halt functions as desired by the user. Memory reference addresses are compared to a preset address. A comparison between the reference and preset address causes an executive trap interrupt to be generated or a program halt to occur, depending on the state of mode control bits. The option includes a register, an 18-bit comparator, an interrupt trigger circuit, and associated control logic.

2-11.2 Query Register

A 21-bit address Query (Q) register is supplied with the Program Halt and Address Trap option. Bits position 17 through 0 contain either the trap address or the program halt address. Bits 23 through 21 are the halt or address trap control bits. When an address is reached in program that coincides with the address stored in the Q register, the machine halts or an interrupt is generated. The Q register may be loaded under program control or via the Programmer's Control Panel.



2-11.3 Operational Description

The Query register is loaded with the Transfer Memory to Query register (TMQ) instruction. This instruction transfers the contents of the selected memory location to the Query register. The 18 least-significant bits, representing the trap or halt address of the memory word, are loaded into bit positions 17-0 of the Q register. Memory word bits 20-18 are not used and bits 23-21 of the memory word are loaded into bit positions 23 through 21 of the Q register. These three bits determine the mode of operation to be performed and have functions as follows:

Bit 23 = ONE	Disable Address Trap
Bit 23 = ZERO	Enable Address Trap
Bit 22 = ONE	Trap on Write only
Bit 22 = ZERO	Trap each time selected address is referenced
Bit 21 = ONE	Trap or Halt during User Mode only
Bit 21 = ZERO	Trap or Halt during Monitor Mode only

When the Program Halt Enable switch on the Programmer's Control Panel is enabled (PH ENA in the up position), the contents of the Q register are compared with the Program address. When they compare, the machine is halted.

When the PH ENA switch on the control panel is disabled, the address trap is enabled or disabled with bit 23 of the Query register. The address trap is enabled when bit 23 is reset, or ZERO. Each time a referenced memory address corresponds with the address stored in the Query register, an executive trap interrupt at level 7 is generated to inform the CPU. When bit 23 is set (ONE), the address trap is disabled. Disabling the trap inhibits the executive trap interrupt.

Additional control of the address trap is provided with bits 22 and 21. With the trap enabled and bit 22 set (ONE), the executive trap interrupt is generated when a write operation is made to the referenced location. If bit 22 is reset (ZERO), the interrupt is triggered whenever the referenced location is accessed.

Memory addresses that result from cycle stealing operations by block controller channels are not affected by the address trap.

2-11.4 Program Control

With the Query register loaded and the address trap enabled, an interrupt is generated (in accordance with the control bit settings) each time a reference is made to the memory location corresponding to the address stored in the Query register. If it is desired that a reference to the selected memory location be recognized only once, a second TMQ instruction should be executed following the first interrupt to set bit 23 of the Q register to a ONE. This disables the address trap.

SECTION III
MEMORY SYSTEM

3-1 GENERAL DESCRIPTION

In the SLASH 6 computer, main memory provides the storage elements for the instruction and data words. Memory modules are available with semiconductor storage elements. Word lengths of 48 bits are stored in the memory modules so that a double word is fetched each time the memory is accessed. The double word so fetched is stored in the memory module's Memory Data register which serves as a two-word Content Addressable Buffer (CAB). When the CPU addresses memory, and the data word is already present in the CAB, the word is made available immediately to the CPU without the necessity of initiating a memory cycle. This scheme serves to reduce the effective cycle time of the computer.

All I/O block controllers communicate directly with the memory modules when performing block mode operations. Full word transfers (24 bits) are made with the IBC and XBC I/O boards. A UBC I/O board transfers double words (48 bits) to and from main memory during a block mode operation.

SECTION IV
INPUT/OUTPUT CHANNELS

4-1 GENERAL DESCRIPTION

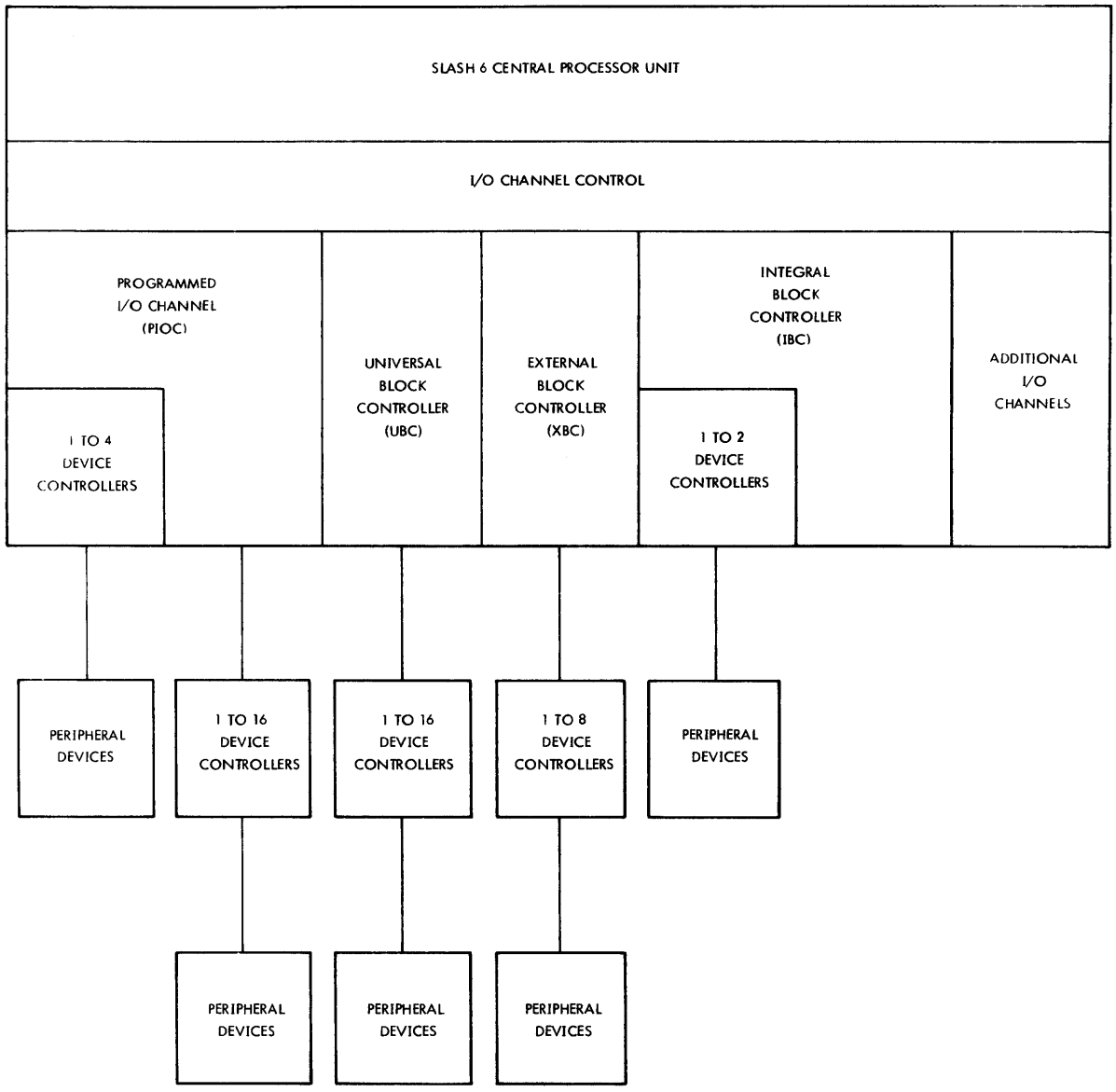
The SLASH 6 Computer Systems input/output (I/O) structure combines the characteristic economy of unit I/O systems with the speed of a channel I/O system. This configuration, in conjunction with the I/O instruction repertoire, permits maximum flexibility in I/O communications. The relationship between the CPU and the I/O structure is illustrated on Figure 4-1. The elements comprising the I/O structure are described in the following paragraphs.

The basic I/O structure allows single word data transfers between the Central Processing Unit (CPU) and a peripheral unit. It also allows I/O command and test operations to be program controlled. Block controller channels may be used to control the transfer of blocks of data between the CPU and the peripheral units without program intervention.

The I/O structure involves communication (such as data transfers, addresses, and command status information) between the CPU and a peripheral unit by way of a channel. The CPU communicates with a specific channel and the channel, in turn, communicates with a peripheral unit. The I/O structure varies with CPU configurations to accommodate an applicable number of Input/Output Channel (IOC) cards, all of which can be active concurrently. Each channel can communicate with from one to 16 peripheral units using standard I/O instructions. Only one peripheral unit per channel can be connected; however, all units can be active at any given time.

Communications between the I/O structure and the CPU may also be conducted on an interrupt basis. Logic in the channel and unit allows unit interrupts to be placed under program control and selectively enabled or disabled by executing the appropriate I/O instruction. An alternate method permits unit functions to be wired directly to the CPU priority interrupt structure and used as interrupt triggers.

The I/O interface is the link between each peripheral unit and its channel. The interface and its associated unit control facilities provide the physical means for connecting the peripheral device to the I/O structure and the logic capability that allows the unit to adapt the standard I/O controls to its specific requirements. The interface facilities and unit control logic are normally integrated with the peripheral unit. However, some controllers are available as options to the Integral Block Channel (IBC).



BD1641-976

Figure 4-1. SLASH 6 Computer I/O Structure Block Diagram

4-2 BASIC I/O CONCEPTS

The I/O structure implements basic concepts to perform input/output operations between the CPU and a variety of channels and units. These basic concepts and their applicability are described in the following paragraphs.

4-2.1 Addressing

a. Channel Addresses - The I/O channels must each be addressed via a unique address contained in each I/O instruction. A channel is patched, or switched, to recognize its assigned address. The recognition of this code in an I/O instruction activates channel logic to execute the instruction. No other channel will respond.

b. Unit Addresses - Since a channel is capable of communicating with two or more unit controllers, any instructions involving the transfer of data, commands, or status must necessarily contain an address applicable to the unit involved. The unit address is contained in the format of the following instructions (Reference: Section VII for formats):

1. Output Command Word (OCW) - PIOC, IBC, UBC, and XBC
2. Output Data Word (ODW) - PIOC, UBC, and XBC
3. Input Data Word (IDW) - PIOC and UBC
4. Input Status Word (ISW) - PIOC, IBC, UBC, and XBC
5. Output Address Word (OAW) - IBC, and XBC
6. Input Address Word (IAW) - IBC
7. Input Parameter Word (IPW) - IBC

NOTE

The inclusion of unit addresses in the IBC channel OAW IAW, and IPW instructions has no transfer-to-unit control. The IBC channel contains the capability to concurrently store block transfer parameters for all unit controllers on its interface and the parameters must be addressed to reserved storage areas.

An instruction containing a unit address, sent to any channel other than the IBC channel is compared to the unit code of the previous instruction. If a non-compare is detected, the channel does not execute the ISW or ODW instruction. Instead, a disconnect/connect sequence is entered in order to connect the addressed unit. A non-compare detected during OCW and ODW instructions forces the disconnect/connect sequence also, but the channel loads the data/command, if not previously set busy, and holds the data/command until the addressed unit is "connected" to its interface.

4-2.2 Disconnect/Connect Sequences

Each IOC performs disconnect/connect sequences if the unit address contained in the instruction differs from the previously loaded address. In disconnect/connect sequences occurring during input instructions, the channel is prevented from setting the "ready" line to the CPU to verify that the instruction was executed. This requires a "Branch on Not Zero" (BNZ) execution after each I/O instruction for a repetition of nonexecuted instructions. Timeout routines sequenced by the CPU may then detect channel/unit hangups and execute Input Status Word (ISW) instructions to pinpoint conditions.

The IBC channel is not equipped to sequence disconnect/connect operations; in this channel the unit is automatically connected to the channel for the purpose of instruction execution except during the time that data transfers are taking place.

4-2.3 Block Controller Channel Priority

A programmable matrix is contained on each IOC capable of performing block transfer operations. The matrix is provided to resolve contention for simultaneous memory cycle requests. The block controller channels are assigned priority levels and the highest priority channel requesting a memory cycle inhibits any lower priority channel(s) from sensing a "memory cycle granted" signal from the CPU. A system should be configured to assign high speed devices a lower priority level than relatively lower speed devices. Also, no unused priority levels should appear between any two channel levels. The priority matrix is patched on UBC and XBC channels and is switch-selectable on the IBC channel.

4-2.4 Handshake (Synchronization) Conditions

With few exceptions, all data and command sequences are synchronized via "handshake" operations. This convention ensures that the connected unit has received the command or data in output transfers or frees the unit to load new words in input transfers. If the unit is unable to accept the command/data, the channel sets itself busy and will honor no output transfer

operation except for the OCW instruction in which "Override" is specified. The normal handshake function is modified in XBC and IBC channel operations and is described following the conventional handshake functions.

4-2.4.1 Output Transfer Handshake

The output transfer handshakes are performed in OCW/ODW single-word transfer operations and in output block transfers of block-controller channels. In single-word transfers, if the channel is not busy executing a previous output instruction, the command/data is loaded into the channel's output buffer and the "Output Command Here" or "Output Data Here" line is raised to the unit. The channel sets itself busy to inhibit any new output transfer operations. When the unit gates the command/data into its own registers, it returns an "Accepted" signal. This signal resets the channel busy condition and the channel is free for a new transfer.

In block transfer sequences, the channel, having been previously initiated for output transfer operations, automatically sequences memory request operations. When the memory cycle is granted, the channel places the transfer address on line and loads the word from specified address. The channel then raises the data transfer handshake line and, when the unit "accepts" the data, fetches another word from memory. The channel remains "busy" and the sequences continue until the transfer is completed.

4-2.4.2 Input Transfer Handshakes

A channel cannot execute an IDW instruction until it senses that the "Data Available" line from the unit has been set true. In normal operations the channel automatically transfers the input to the CPU and raises the "Data Accepted" handshake line. The unit drops "Data Available" to prepare a new word for transfer.

An input block transfer begins when a unit raises its "Data Available" line after the channel and unit have been commanded to the input mode. The channel loads the data into its input buffer and raises its "Data Accepted" line to the Unit. The channel then sequences a memory cycle with the CPU to load the input word at the address specified by the Transfer Address Register (TAR). The channel will not honor any subsequent load requests until the memory cycle has been completed.

4-2.4.3 XBC Channel Handshakes

The block transfer sequence control is under the control of the external units in XBC applications. The unit may be commanded to the block-transfer mode via an OCW instruction and may require parameter inputs but, once initiated, the device controls the transfers. In executing the OCW instruction the channel uses the conventional CDH handshake signal and the unit returns "Accepted" to signal loading of the command. If required by the unit, the channel executes an OAW instruction to provide the Transfer Address (TA) to the unit. The channel raises "Address Word Here" which signals the unit to "accept" the address. This may be followed by an ODW instruction in which the word count is sent to the unit. The channel raises "Word Count Here" which the unit "accepts."

Data transfers to/from memory begin when the unit sets a priority-structured "Data Transfer Request" line to the channel. If the channel is not busy executing an instruction or servicing a higher-priority request, the channel raises its "Send" line. The unit responds via its "Ready" line. The unit then places the TA on line for channel storage and sets a transfer direction control line, the "In" line. If the "In" line is received in its true state, the channel loads the data from the unit, sets itself busy, and requests a memory cycle for setting the data into memory. When the "In" line is received set false, the channel requests a memory cycle for access purposes and, when the cycle is granted, the channel loads the data word from the address furnished by the unit. The channel then pulses its "Output Data Here" line to load the data into the unit.

4-2.4.4 IBC Channel Handshakes

The IBC channel is sequenced for block transfers via the units' "Data Transfer Request" lines. (See previous paragraph for similar transfer capability.) The channel also specifies the transfer direction, but this is a reflection of the command word to the unit. In normal operation, channel parameters are loaded via the conventional block-transfer-initiate sequences into RAM locations reserved for units served by the channel. The unit, however, may be commanded to an external addressing mode in which its loads the unit's TAR and controls whether the TAR and/or Word Count Register (WCR) are incremented/decremented, respectively.

The IBC channel does not "shake hands" with the unit during command transfers; the command is automatically loaded by the unit controller since the channel "selects" the unit, bypassing the usual disconnect/connect sequence.

4-2.5 Timing

All of the I/O channels except the IBC depend solely on computer clock pulses for execution of single-word instructions or, where applicable, block-transfer operations. The IBC channels are synchronized to CPU timing for some sequences but may provide other sequences via independent "internal" timing.

4-2.6 Block Transfer Memory Access

Block controller operations are controlled by the channel after it has been initiated under program control. The channel, therefore, accesses memory for read/write operations and must request memory cycles for this purpose. In memory transfers, the requested memory cycle is automatically granted unless the CPU is in the C2 cyle of the following instructions:

TBM	TXM	AAM	ZBM
AMB	TMX	AUM	THM
SMB	IMI	TMO	
DMX	IME	TDM	
MMX	IMA	USP	
SMX	AIM	AOM	
AMX	AEM	FBM	

NOTE

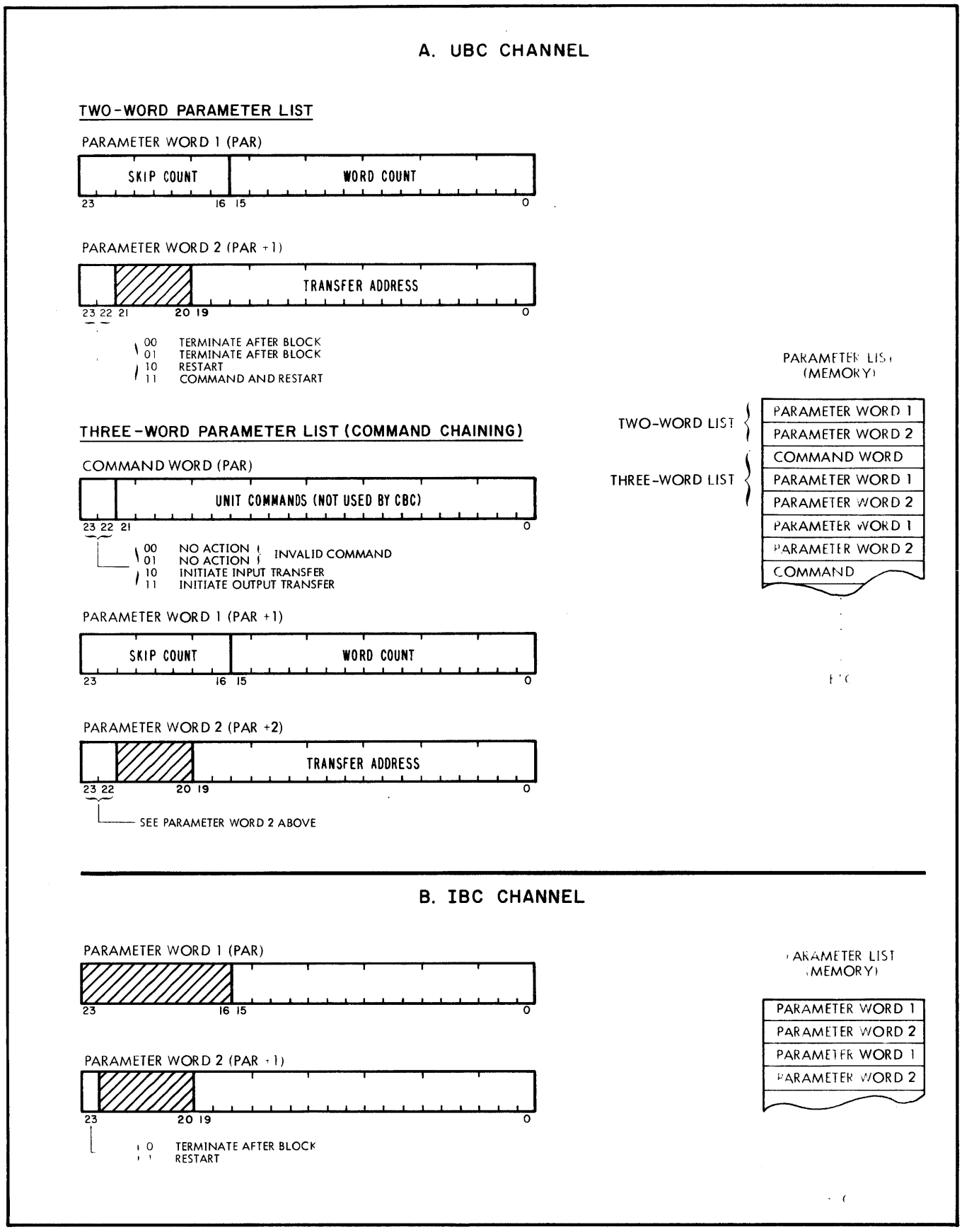
An interrupt will delay a grant operation by one cycle.

4-2.7 Block Transfer Parameters

The UBC and IBC are initiated for block-transfer operation via an OCW instruction. The command word itself must have its bit 23 set to activate the block-transfer mode. These conditions activate the channel to sequence two simultaneous memory requests for parameters. The designated parameter words are illustrated in Figure 4-2.

4-2.7.1 UBC Channel Parameter Words

The UBC channel parameter word formats are illustrated in Figure 4-2. In this channel the OAW instruction preceding the OCW used to initiate the block transfer control causes the first parameter address (PA) to be loaded into a parameter address register (PAR). This allows the parameters to be located in a separate "List", but the list must be located in the lower 65K of memory. Each time the PAR is addressed for a parameter word, the channel increments the PAR for subsequent parameters.



MI1821-976A

Figure 4-2. Block Controller Parameter Word Formats

The first parameter applicable to UBC operations contains a 16-bit word count and the most-significant 8 bits contain a "Skip Count". The skip count is significant only in block transfers designated for input and is loaded into the channel's skip count register (SCTR). This parameter controls the actual transfer operations in which data is loaded into memory. When a count is set into the SCTR, the channel provides load sequences to transfer the data from the unit to the channel but does not request memory cycles to load the data into memory. The SCTR is decremented with each word transferred and, when the counter has decremented-to-zero, the channel begins data transfers to memory based on the word count parameter.

The second parameter word in UBC applications contains an 18-bit TA. The two most-significant bits of PW2 are stored in the channel and specify three termination sequences that may be entered when the block transfer has been completed; these are:

a. Normal termination - the channel goes to an idle state when the last data word has been transferred.

b. Data restart - the channel goes into a re-initiate sequence to bring in two new parameters. The subsequent block transfer is as specified in the OCW initiating the previous block transfer.

c. Chain command restart - the channel goes into a re-initiate sequence in which a new command (from memory) is sent to the unit to change the transfer direction. As with the OCW initiating block mode operations, bit 23 of the command word must be set to command the initiate sequence. This is followed by bringing in PW1 and PW2 to set channel control action for the block of data to follow.

4-2.7.2 XBC Channel Parameter Words

The XBC channel does not contain circuits to store and control parameters. Likewise, the channel does not provide any restart actions. The parameters are controlled by the external device, but the device may require that the parameters be initially furnished from memory. In the latter case, the channel is sequenced to execute an OAW instruction which transfers the TA to the unit. This is followed by an ODW instruction which sends the word count to the unit. After being initiated by the OCW command, each data transfer is sequenced and the unit itself furnishes the transfer address. The unit controls the word count and generates any operational interrupts.

4-2.7.3 IBC Channel Parameters Words

The IBC channel is initiated to the block-transfer mode via the conventional OCW with bit 23 of the command word set. The IBC channel then enters the initiate sequence to load two parameter words (Figure 4-2). The first parameter word contains the word count of the subsequent block transfer. The second parameter word contains an 18-bit TA and the "restart" condition. The IBC channel does not provide Chain Command functions in a restart operation. But, since the IBC channel contains storage for parameter addresses, the channel may access PW1 and PW2 from a "list".

4-3 INPUT/OUTPUT INSTRUCTIONS

Execution of I/O instructions consists of the transfer of command (OCW), data (ODW and IDW), status (ISW), or address (OAW, IAW, IPW) words between the A register and the specified channel/unit combination. The channel/unit codes in each I/O instruction (exclusive OAW, IAW, and IPW instructions in applicable block-transfer channels except the IBC IOC) allows one channel to be selected and one of up-to-16 units to be connected to the channel. When an instruction to the same channel carries a different unit code, the previously-specified unit is disconnected and the new unit is connected automatically. During this disconnect/connect sequence, the channel is busy and does not respond to I/O instructions until the sequence is completed. If a channel is in the process of transferring commands or data to a unit, an ISW or IDW instruction addressed to a different unit on the same channel receives a busy indication.

Command and data words from the CPU are transferred to the channel output buffer and subsequently to the connected peripheral unit. Data and status words are retained in the input buffer of the selected unit and transferred to the A register upon request (instruction) from the CPU. Address words are applicable only to those channels employing the block-control capability. (Refer to I/O instruction formats in Section VII for the following discussions).

4-3.1 I/O Commands

The OCW instruction transfers a command word to the specified channel/unit combination. The command word bits specify the unit control function(s) to be performed and/or the I/O condition to be established. Following the execution of an OCW instruction, the channel remains busy until the command has been accepted by the addressed unit. Figure 4-3 shows the format for a typical OCW instruction.

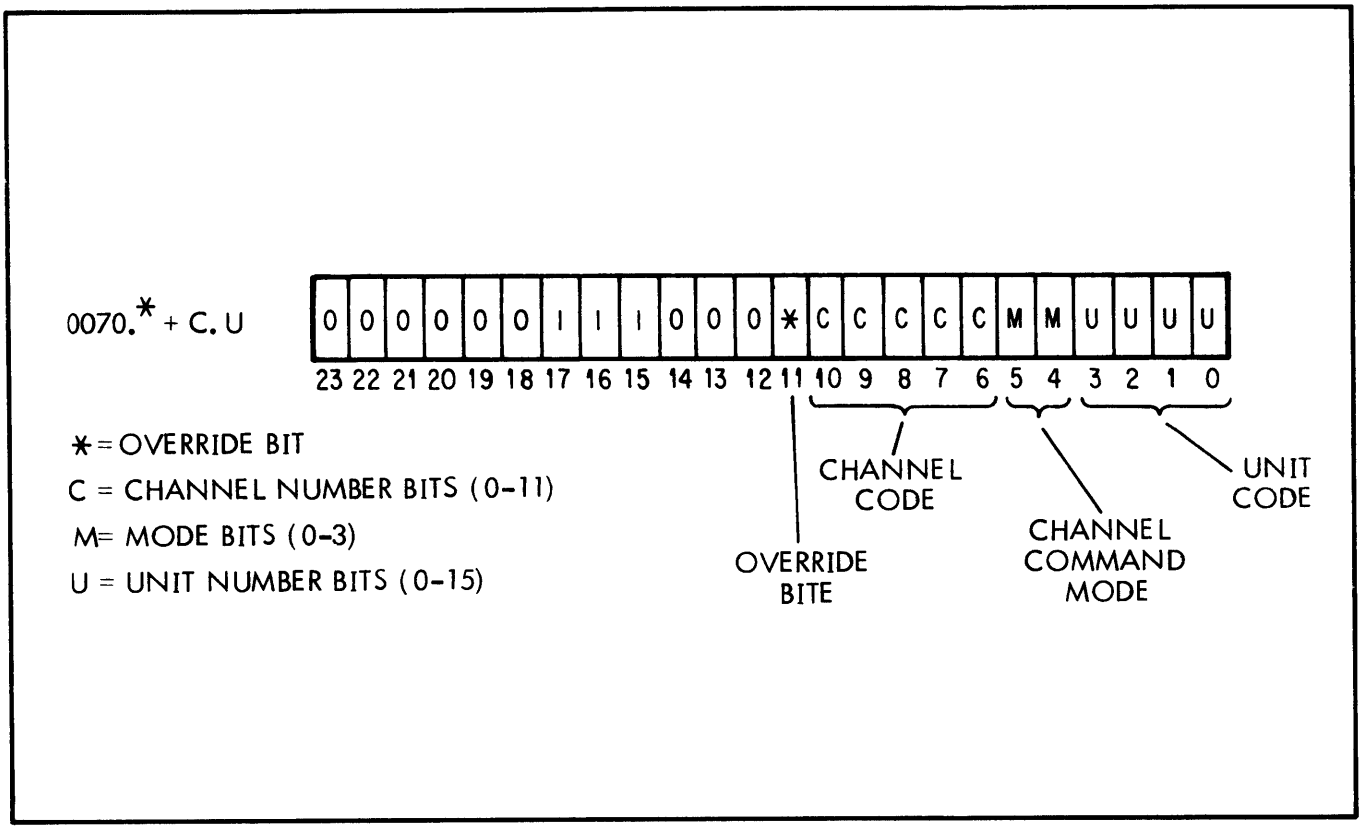


Figure 4-3. OCW Instruction Format

If the channel is busy or not ready when addressed by the OCW instruction, the Condition register is set to "Not Zero" to allow a programmed delay. The Override function causes the channel to go through a unit disconnect/connect sequence regardless of the unit code. This clears the channel/unit of any other activity and allows the current instruction to assume control of the channel unconditionally upon termination of the disconnect/connect sequence.

All of the I/O channel execute the OCW instruction, but channel capabilities may require setting of the instruction contents as follows:

a. Unit Addresses - The IBC channel contains interface capability for up-to-two devices. The unit address must therefore be set in Unit Code bits 0 and 1. The unit addressing requirements for the XBC channel is contained in Unit Code bits 0-2. All of the remaining channels, having the capability to interface with up-to-16 units, utilize all of the Unit Code bits for addressing purposes.

b. Channel Command Mode - Bits 4 and 5 provide command control to set an I/O channel to one of four modes: Normal, Offline, Multiplex, and Reset. The Normal mode specifies "normal" command functions. The Offline mode removes the units from the I/O channel interface, permitting a second computer and I/O

channel to assume control over the units. The Multiplex mode allows a "Master" unit to communicate with a "Slave" unit and the CPU cannot intervene except via a Master Clear or an OCW instruction with "Override" specified or Reset mode commanded. The Reset mode allows a return to Normal mode operations from either the Offline or Multiplex modes.

The IBC channel does not respond to the mode control specifications of an OCW instruction (they thus always operate in the "Normal" mode).

The XBC, and IBC channels may be commanded to any of the modes described above.

c. Override Control - This OCW instruction control function is exercised in all I/O channels except the XBC. An OCW instruction with this bit set assumes immediate control of the channel/unit by forcing a disconnect/connect sequence.

4-3.2 I/O Status Word

The ISW instruction is used to test the operational status of the channel/unit. When a channel is addressed by the ISW instruction, a 24-bit status word is transferred to the A register in the CPU. The quantity and significance of the status bits depends on the type of peripheral unit involved. Units controlled by 8-bit interface channels (e.g., PIOC) furnish up to six unit-defined status bits which the channel sets into the least-significant bits of the input word. Channels with 24-bit unit interfaces (e.g., block controllers) may receive as many as 8 unit-defined status bits which are set into the 8 least-significant bits of the input word.

Channel status may be set into the three most-significant bits of the input word and reflect the channel's current mode or "busy" status as follows:

IBC	None
PIOC	Bit 21 - Multiplex
	Bit 22 - Offline
UBC	Bit 21 - Multiplex
	Bit 22 - Offline
	Bit 23 - Busy (Block Transfer)
XBC	Bit 22 - Offline

4-3.3 Single Word Data Transfers

A. Input Data Word

The IDW instruction is a request from the CPU to a specific channel/unit combination for a data word. If data is available, the data word is transferred immediately to the A register.

If data is not available, the Condition register is set to "Not Zero" to allow a programmed delay.

Normally, the 24-bit input data word contains a single data character. The actual number of data bits per character depends on the channel and unit involved in the transfer. For example, the console typewriter generates an 8-bit character and a card reader may generate a 12-bit character. In any case, the character is right-justified in the A register with the unused bit positions set to ZEROs.

Assuming the data character contains no more than 12 bits, more than one character may be packed in the A register through the use of the Merge feature. When a character Merge is employed, a logical OR is performed between the previous contents of the A register and the new input data word. Without the Merge, the previous contents of A are destroyed upon transfer of a new character to A. An illustration of the character Merge technique, as compared to a normal IDW instruction, is shown in Figure 4-4.

The IDW instruction is executed by all I/O channels except the XBC and IBC.

B. Output Data Word

When an ODW instruction is executed, an 8- or 24-bit data word is transferred from the A register to the specified channel. The data word is subsequently transferred from the channel to the unit that is currently connected. If the channel is busy or not ready to accept the data word, the Condition register is set to "Not Zero" to allow a programmed delay. If the unit is not ready to accept the data from the channel, the data remains in the channel buffer.

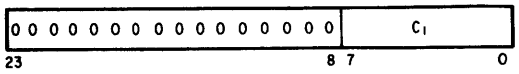
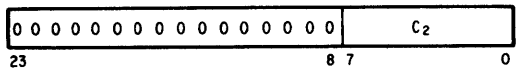
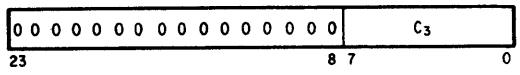
As soon as the peripheral unit is able to accept the data from the channel, the channel-to-unit transfer is made, thereby freeing the channel buffer for another data (or command) word from the CPU.

The number of data bits accepted by the peripheral unit varies according to the type of unit involved. Some peripheral units are word-oriented and accept the entire 24-bit word. Others are character-oriented and accept only a specific number of bits per character.

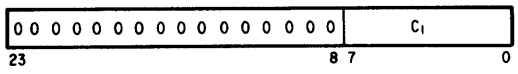
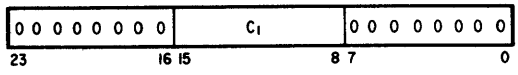
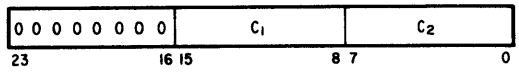
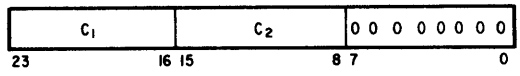
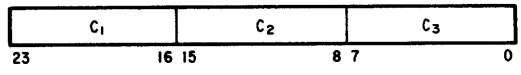
The ODW instruction function in XBC I/O channels serves the purpose of sending a word count parameter from the CPU A register to the addressed unit, if required by the unit. In subsequent block-transfer operations the unit controls the WC parameter. The IBC channel does not execute the ODW instruction.

EXAMPLE: THREE 8-BIT DATA CHARACTERS ARE TO BE PACKED IN THE A REGISTER.

(a) NORMAL (WITHOUT MERGE)

CODING	COMMENTS	REGISTER A
IDW CU	BRING IN FIRST DATA CHARACTER	
. . .		
IDW CU	BRING IN SECOND CHARACTER	
. . .		
IDW CU	BRING IN THIRD CHARACTER	
. . .		

(b) MERGE

CODING	COMMENTS	REGISTER A
IDW CU	BRING IN FIRST DATA CHARACTER	
LLA 8	SHIFT LEFT 8 PLACES	
IDW* CU	BRING IN SECOND CHARACTER AND MERGE	
LLA 8	SHIFT LEFT 8 PLACES	
IDW* CU	BRING IN THIRD CHARACTER AND MERGE	
. .		

MI60-005-770

Figure 4-4. IDW Instruction; Data Character Formatting

4-3.4 Address Transfers

Three address-transfer instructions are executed by block-transfer channels for the purpose of channel or unit set-up for subsequent transfers (OAW) or for CPU checks of transfers progress (IAW and IPW). However, the PIOC channel, may execute the OAW instructions. The following discussions cover applicability and qualifications for the address-transfer instructions.

A. Output Address Word

The OAW instruction is executed by the UBC and IBC to set the starting address of parameters for block-transfer control. (Reference: Paragraph 4-2.7 for control actions.) The XBC also executes the OAW instruction, if a unit on its interface requires a TA starting address.

The IBC and UBC, channels load their respective PAR during execution of the OAW instruction. The instruction is executed in a single machine cycle.

NOTE

In UBC execution of the OAW instruction, the block-transfer logic is cleared. Therefore, this instruction should not be programmed for execution until all block transfer operations are completed.

The XBC channel will not execute the OAW instruction if the channel is busy executing an output instruction or a data transfer. The instruction word must be addressed to the unit to which the TA parameter is intended. Therefore, a "programmed delay" should be programmed to facilitate instruction execution.

In IBC channel OAW execution the instruction word must be addressed to a unit controller contained on the channel card. The channel executes the instruction in a single machine cycle, writing the PA into a register reserved for the addressed unit.

Available for software interrupt purposes, an Interrupt Generator is located on the PIOC channel to allow generating one-of-four possible interrupt pulses in response to an OAW instruction. The instruction is executed automatically by the addressed channel to provide one microsecond interrupt pulses.

B. Input Address Word

The IAW instruction may be addressed to any of the block-controller channels except the XBC channel. For IBC channel purposes the instruction word must be addressed to the channel and unit; otherwise the instruction is addressed only to the desired channel. In all applicable channels except the IBC the instruction is automatically executed during the current instruction

cycle. The IBC channel executes the instruction only if it is not busy executing another instruction or transferring data. In all cases, the channel sets its "Ready" line to the CPU to clear the C register. The address word is sent to the A register and may be used as a check on transfer progress. The word represents the TA of the current transfer and is always 18 bits wide.

C. Input Parameter Word

This instruction is very similar to the IAW. The instruction is addressed only to those block-controller channels capable of PA storage: UBC and IBC channels. The execution of the IPW instruction is identical to the IAW instruction. However, the UBC channel is limited to a 16-bit parameter address.

4-4 INTERRUPT CONTROL

The OCW instruction may be used to selectively enable and disable two peripheral unit interrupts. The two interrupts are defined as Input and Output and are controlled by bits 0-2 of the command word. Table 4-1 illustrates the various bit configurations.

The terms "input interrupt" and "output interrupt" are applicable only to peripheral units that are equipped with both input and output data handling facilities. Input-only devices may make use of the input interrupt and an alternate interrupt at the normal output level. Output only devices may make use of the output interrupt plus an alternate at the normal input level.

When the unit input interrupt has been previously enabled, an Input Interrupt signal will be generated when the input buffer in the unit is loaded (i.e., the same time the "Data Available" signal is generated). An I/O channel has no control over an input interrupt.

Table 4-1. Peripheral Unit Interrupt Control

Command Word Bit Configuration	Action
2 1 0*	
0 0 0	No Action.
0 0 1	No Action.
0 1 0	Disable Input (or Alternate) Interrupt
0 1 1	Enable Input (or Alternate) Interrupt
1 0 0	Disable Output (or Alternate) Interrupt
1 0 1	Enable Output (or Alternate) Interrupt
1 1 0	Disable Both Interrupts
1 1 1	Enable Both Interrupts

*No significance to some units, i.e., the interrupts are unconditionally enabled by CW Bits 1 and/or 2.

When the unit "output interrupt" has been previously enabled, an Output Interrupt signal may be generated by the channel for two sets of conditions based on a device-defined signal, "Enable Channel Buffer Empty Interrupt" (ECBEI). If the unit raises ECBEI to the channel, the Ouput Interrupt will be generated for a minimum of 325 nanoseconds if:

A. PIOC channel;

1. the channel has not been commanded to the Offline or Multiplex mode, and,
2. the channel is not performing a disconnect/connect sequence, and,
3. the channel's output buffer is not holding a command/data word for unit transfer purposes.

B. UBC channel;

1. the channel has not been commanded to the Offline mode, and,
2. the channel has not been manually set to respond only to Offline commands, and,
3. the channel's output buffer is not holding a single-word output command/data word, or is not holding a block-transfer output data word.

C. XBC and IBC channels;

These channels contain no Output Interrupt capability.

If the unit holds the ECBEI signal to the channel low, the Output Interrupt will be generated by the channel but the channel's output buffer condition (3, above) is ignored. Instead the device-defined state of Status Bit 2 from the unit is allowed to set the Output Interrupt. The mode and manual conditions described for each type of channel above remain in effect.

4-4.1 Block Transfer Initiate Interrupts

The UBC channel may generate an "Initiate Interrupt" under various conditions. Normally the interrupt is generated by the channel in response to an OAW instruction but alternate conditions may provide the interrupt. The two general conditions available are:

A. OAW Instruction

The channel is commanded to execute an OAW instruction, has not been patched to disable the interrupt generating capability, and has not been set to the Offline or Multiplex modes by Software or manual (switch) actions.

B. Multiplex Mode

The channel has been commanded to the Multiplex mode and has not been patched to disable this interrupt generating capability.

In the OAW instruction conditions, the UBC channel pulses the Initiate Interrupt line for approximately 500 nanoseconds.

In the Multiplex mode conditions, the Initiate Interrupt line is set true when the channel enters the mode and remains true until the channel is removed from this mode of operation.

4-4.2 Block Transfer Word Count Complete Interrupt

The UBC channel contain the capability to generate a " Word Count Complete" interrupt when the channel has loaded the final word of a block-transfer operation, if the channel has not been commanded to the Offline or Multiplex modes. (The IBC channel generates a " Word Count Complete" signal to the unit when the channel has loaded the final word, if no " Restart" is specified. This signal, however, is under the control of the unit for interrupt purposes). The approximate durations of the interrupt is 475 nanoseconds.

4-5 I/O CHANNEL SWITCH/PATCH CONTROLS

The various I/O channels contain switch and patching provisions to perform a number of operational functions. The PIOC channel's patching capabilities is restricted to channel address selection (Paragraph 4-2.1). The block-transfer channels are also patched, or switches set, to encode a unique channel address, but those channels also contain a variety of other manually-activated functions. These functions are listed in Table 4-2 with I/O channel applicability specified.

4-6 I/O CHANNEL OPERATIONAL SUMMARIES

The following paragraphs summarize single-word and block-mode transfer capabilities of the various I/O channels interfacing with the SLASH 6. Included are program lists and suggestions. Refer to Paragraph 4-3 for application to specific I/O channels.

4-6.1 Single-Word Instructions Execution

A. OCW/ODW

The channel, if not busy, loads a command/data word from the CPU A register into its output buffer. The channel sets itself " busy" to inhibit any further instruction executions until

it has completed the transfer to the addressed unit. In the event of a disconnect/connect sequence, the channel withholds the handshake until the addressed unit is "connected" to its interface. A BNZ instruction should be programmed to verify channel execution of the OCW instruction.

B. IDW

The channel executes this instruction in one machine cycle, if the channel is not busy executing an output transfer, is not involved in a disconnect/sequence, and the connected unit has signalled that data is available for transfer via its "Data Available" line. The BNZ execution performed by the CPU provides verification of transfer. The channel shakes hands with the connection unit and is ready for further instructions.

C. ISW

An I/O channel executes this instruction if the channel is not busy executing an output transfer and is "connected" to the addressed unit.

D. OAW

This instruction is addressed to block-controller channels (channel and unit in XBC/IBC applications) for the purpose of transferring the address of the first word involved in control of a subsequent block data transfer. Channel loading of the output word from the CPU's A register into the channel's PAR (UBC applications) is automatic. In XBC applications the instruction involves transferring a TA to the unit for subsequent control by the unit. The XBC channel must have gone to "not busy" prior to instruction time for execution. A programmed delay must therefore be executed by the CPU for verification of transfer. A handshake with the unit is performed in this instruction and the channel sets itself busy until the transfer-to-unit is completed. In IBC applications the addressed channel executes the instruction unless previously set busy via an instruction or data transfer sequence.

E. IAW/IPW

The IAW/IPW instructions are executed to transfer the contents of a block-controller channel's TAR/PAR to the CPU's A register. The IPW and IAW instructions are not executed by XBC channels. The instructions, when applicable, are executed automatically in UBC channels. The IBC channel is inhibited from executing the instruction if currently busy in an instruction or data transfer operation.

4-6.2 Block-Transfer Operations

All block-control channels are initiated by computer control for block-transfer operations and proceed under self control or unit control to perform the transfer operations. The following paragraphs describe general performance of block transfers applicable to each channel. Refer to Figures 4-5 through 4-7 for simplified flow diagram of block-transfer operations.

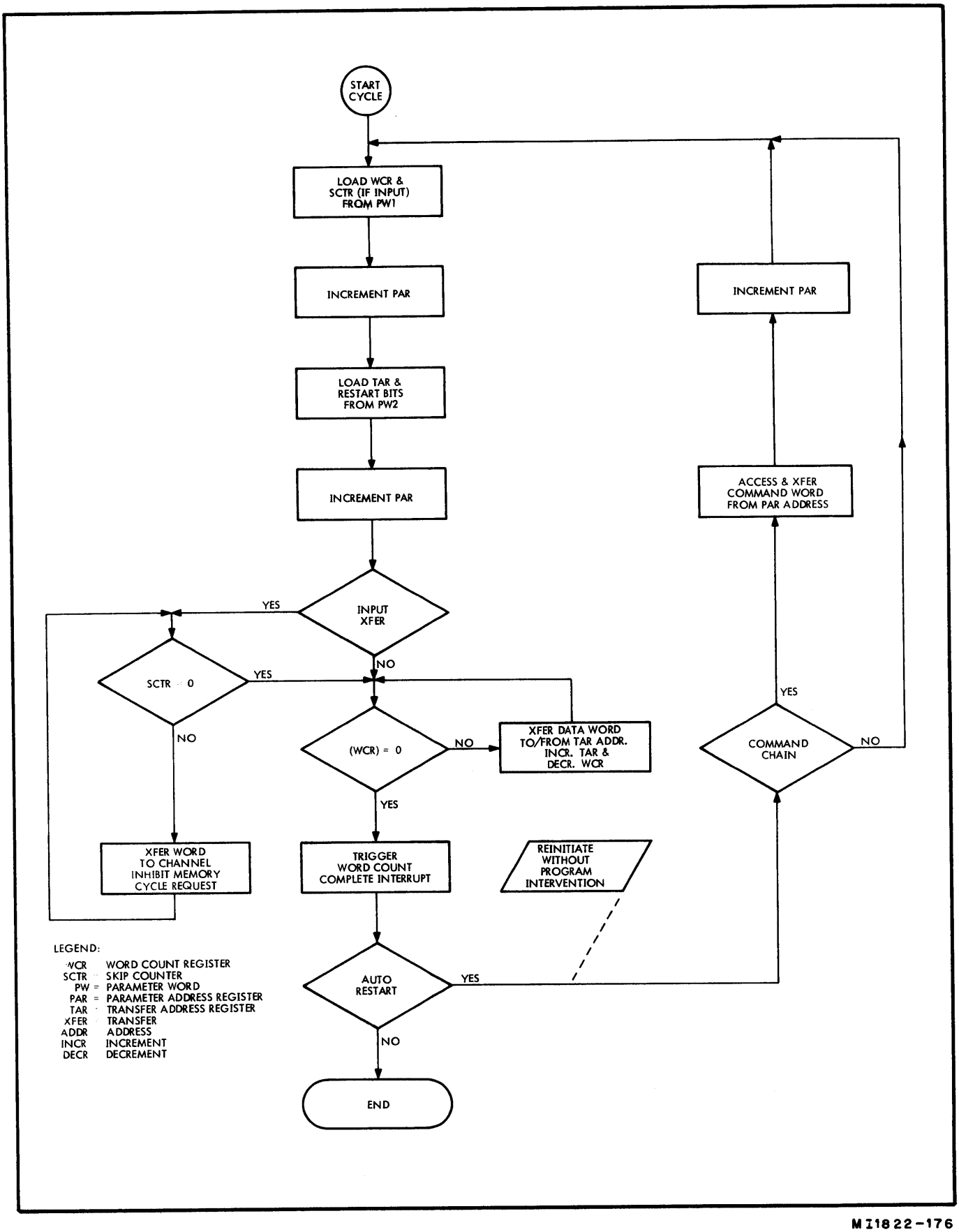
Table 4-2. I/O channels Manual Control Capabilities

Function	Channel Type			
	PIOC	IBC	UBC	XBC
Permanent Offline/Multiplex mode selection	Switch		Switch	
Channel code selection	Switch	Switch	Switch	Switch
Memory Cycle Priority		Switch	Patch	Patch
Unit selection		Patch		

4-6.2.1 UBC Channel Block Transfers

The UBC channel is "set-up" via an OAW instruction and initiated via an OCW instruction with Bit 23 of the unit command specifying the block transfer and Bit 22 specifying the direction of transfer. During the OCW sequence the channel sets itself "busy" to all ODW, IDW, and OCW instructions (except an OCW specifying "Override"). The channel remains busy for the duration of transfers initiated by the OCW instruction. The channel automatically loads two parameter words (see Figure 4-2) via "cycle-stealing" conventions (Paragraph 4-2.5). If the output transfer has been specified, the channel sequences a memory request and specifies the location via its TAR. The channel increments the TAR, decrements its WCR, and loads the data word in its output buffer when the memory cycle is granted. The channel then "shakes hands" with the unit to complete the transfer. The channel then fetches another word for transfer. When the WCR has decremented to ZERO, the channel examines its "Restart" parameter (Bit 23 of PW2) and either re-initiates itself for another block transfer or returns to an "idle" state, resetting its "busy" condition.

If the input transfer was specified via the OCW, the channel waits for the unit to signal data availability. The channel then loads the input data into its input buffer and signals "accepted" to the unit to free it for the next word. The channel then requests a memory cycle, and, when granted, places the TA and data on line to memory. The channel increments the TAR, decrements the WCR, and returns to sense the unit's "Data Available" line. This sequence continues until the WCR forces the restart sequence as described above.



MI1822-176

Figure 4-5. UBC Block Transfer Sequence; Simplified Flow Diagram

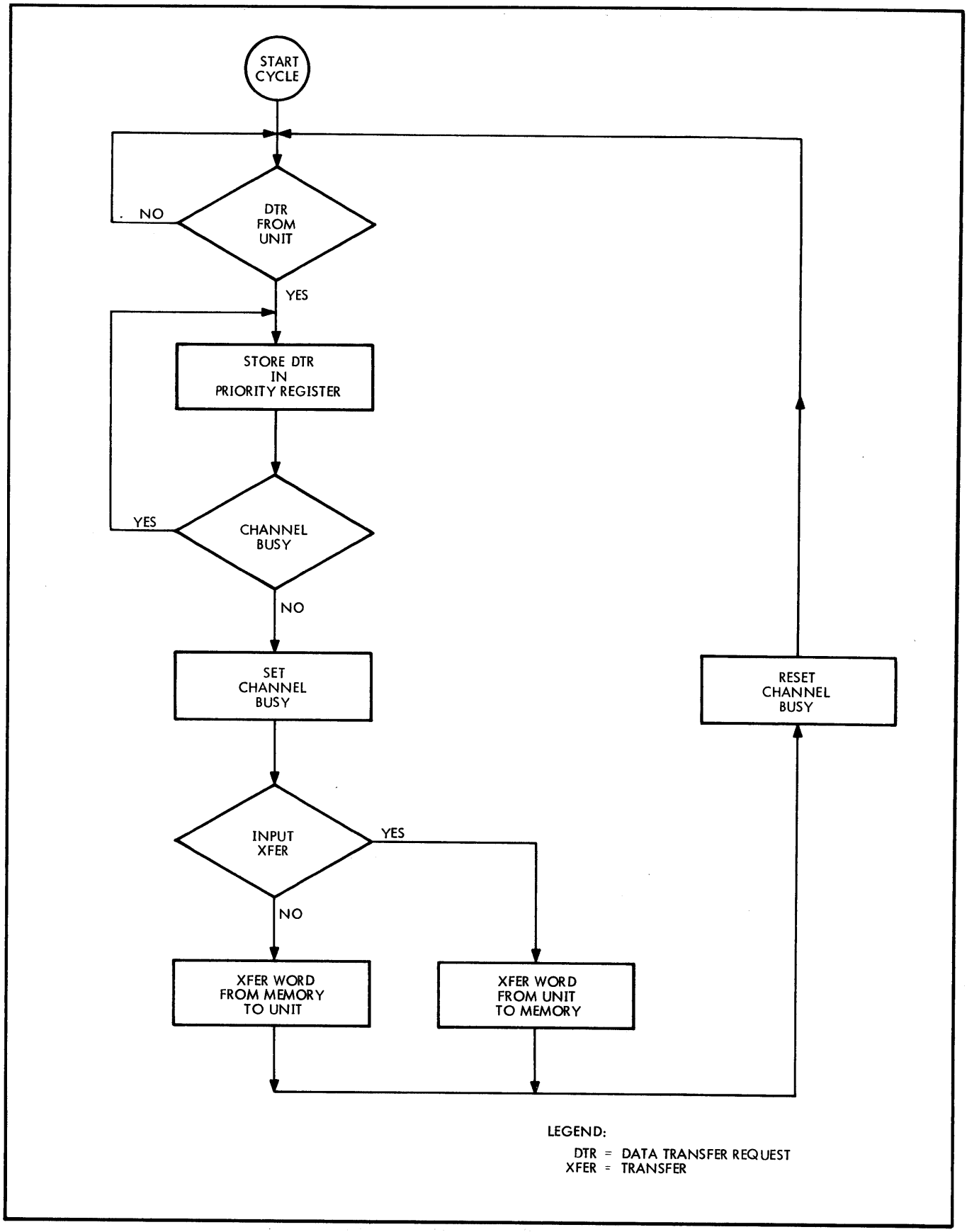
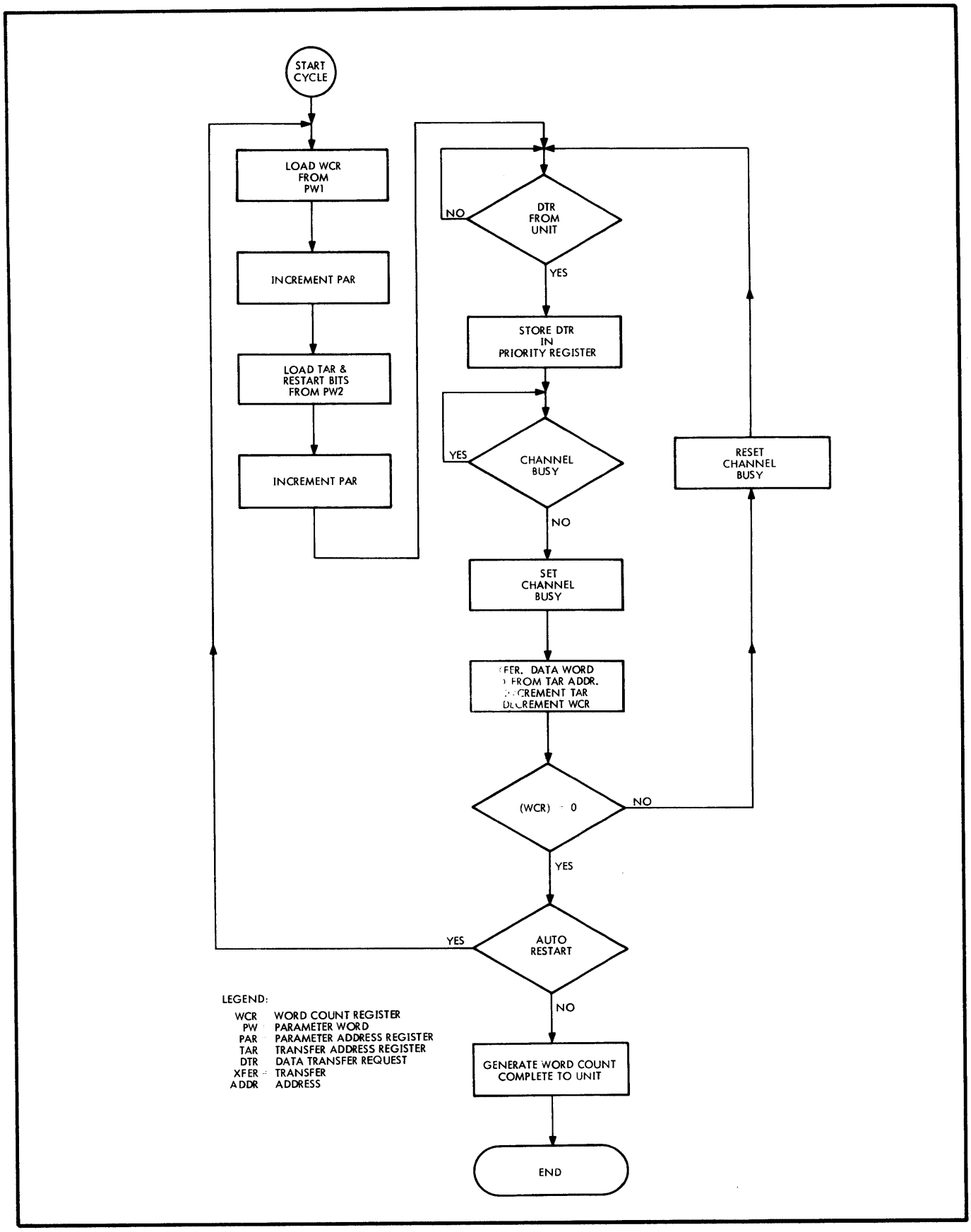


Figure 4-6. XBC Channel Block Transfer Sequence; Simplified Flow Diagram



MI1824-176

Figure 4-7. IBC Channel Block Transfer Sequence; Simplified Flow Diagram

The UBC channel contains a Skip Count Register for added parameter control in input transfer operations and may enter an alternate "Restart" after a block of data has been transferred. (See Figure 4-2 for UBC parameters).

The output data transfers are sequenced in an identical fashion. The channel's capability to "Restart and Chain Command" allows the Re-initiate sequence to access an additional parameter (in this application, a new command to the unit) to change transfer direction without program intervention. In this situation, the new command word initiates the channel in the same manner as did the original OCW instruction.

The Skip Counter affects only those transfers slated for memory. The skip count allows the channel to pass over unwanted data (sync codes, etc.) before actual data loading is sequenced. When the skip count parameter specifies a count, the channel sequences handshakes with the unit to unload the unit, but the channel does not request the memory cycles from the CPU to load the data into memory. The SCTR is decremented with each transfer, but the TAR and WCR remain unchanged. When the SCTR has decremented to ZERO, the channel begins loading data words into memory using "cycle stealing" conventions.

4-6.2.2 XBC Channel Block Transfers

The XBC channel is normally initiated to block-transfer operations via an OCW instruction in which a command is transferred to the unit. If required, the OCW may have been preceded by OAW and/or ODW instruction to transfer TA and WC parameters to the unit. Once initiated, the channel is under the control of the unit for transfer purposes. When the unit signals a "Data Transfer Request" (DTR), the channel, if not previously set busy, sets itself busy, and stores the TA from the unit. The unit specifies the transfer direction and, if an input transfer is specified, "accepts" the data from the unit. The channel then requests a memory cycle and, when granted, transfers the data to memory, based on the TA furnished by the unit.

If the unit specifies an output transfer, the channel requests a memory cycle. When the cycle is granted, the channel places the TA on line to memory, stores the data from memory and performs a "Data Here"/"Accepted" handshake with the unit in which the data is transferred to memory.

The XBC channel's "busy" condition is reset after each instruction or data transfer is accomplished. The unit controls the TA and WC parameters and generates any required interrupts. The maximum transfer rates for XBC channel block-transfer operations are 444,000 words/seconds (Output) and 444,000 words/second (Input).

4-6.2.3 IBC Channel Block Transfers

The IBC is set-up and initiated for block transfers via the OAW and OCW instructions, but the channel sets itself "not busy" after each instruction or data transfer. The channel may thus store the two parameter words for up-to-two self-contained unit controllers (Figure 4-2) and interleave data transfer.

Data transfers are sequenced by the channel based on "Data Transfer Request" signals from the units. The DTR lines are priority-structured and the unit indicates the direction of transfer. Data transfers then proceed as described for XBC channel operations except as follows:

A. the unit allows/inhibits TA and WC incrementing and decrementing by setting its "Block Mode" control line true/false (See External Addressing Mode below).

B. the unit does not furnish the TA parameter (except in the External Addressing mode).

C. the channel/unit does not "shake hands" in output transfers.

D. the channel generates "Word Count Complete" to the unit, only, which then controls the interrupt to the CPU.

The IBC channel may enter an External Addressing mode by presenting its DTR, "Address Here," and "Input" lines set to the channel. The address is then loaded into the TAR for the specified unit. The data presented with the next DTR is transferred into or out, as set, of the memory address of the unit's TAR. If the "Address Here" signal is not presented again to change the TAR, any further data transfers will use the same TAR address. This allows the use of a specified memory address as a register.

The maximum transfer rates for the IBC channel block-transfer operations are 444,000 words/second (Output) and 666,000 words/second (Input).

4-6.3 Program Lists

The following program lists specify various Software control functions for block-transfer I/O channels. Note the functional identify of the applicable channels.

4-6.3.1 IBC Channel Applications

The following examples illustrate three different IBC applications.

Example 1: Simple, single buffer input.

```
        TOA     PA      Parameter Address
        OAW     C       Initialize TAR
        TMA     CW      Command Word
        OCW     CU      Initiate transfer
        BNZ     *-1     Delay if channel is busy
        . . .
CW      DATA            Bit 23 and others as required by the I/O device
PA      DAC             Absolute Word Count
        DAC     BUFF    Address of Input buffer
BUFF    BLOK    n       Reserve n words. Word n+1 is of no significance
                        since the AR bit is not set.
```

Example 2: Multi-buffered output with automatic restart and buffer switching.

```
        TOA     PA1     Parameter Address 1
        OAW     C       Initialize TAR
        TMA     CW      Command Word
        OCW     CU      Initiate first transfer
        BNZ     *-1     Delay if channel is busy
        . . .
CW      DATA            Bits 23, 22, and others as required by the I/O
                        device.
PA1     DAC     n       Word Count
        DAC*    BUF1    Address of buffer 1 and the ARF (*)
PA2     DAC     n       Word Count
        DAC*    BUF2    Address of buffer 2 and theARF (*)
BUF1    BLOK    n       Reserve n words
        DAC     PA2     Automatic Reinitialization address for TAR,
                        to switch buffers
BUF2    BLOK    n       Reserve n words
        DAC     PA1     Automatic reinitialization address for TAR,
                        to switch buffers
```

NOTE

Once this cycle is initiated it will continue, without program intervention, until a new command is received.

Example 3: Multi-buffer input with automatic buffer switching but without Automatic Restart.

```
        TOA     PA1     Parameter Address 1
        OAW     C       Initialize TAR
        TOI     BUF1    Setup pointer for buffer to be processed
        TIM     BA
        TOI     BUF2    Prepare for initial interchange
        TMA     CW      Command Word
        OCW     CU      Initiate first transfer
        BNZ     *-1     Wait until channel takes command
SBAR    IMI     BA      Switch process pointer
        TMA     CW      Command Word
        OCW     CU      Initiate next transfer
        BNZ     *-1     As soon as channel is free process current
        . . .           buffer, whose base address is in register I,
        . . .           while the alternate buffer is being filled.
        . . .
        BUC     SBAR    To switch buffers and restart TBC
        . . .
CW      DATA            Bits 15, 14 and others required by I/O device
BA      ZZZ             Alternating buffer address
PA1     DAC     n       Word count only
        DAC     BUF1    Buffer 1 address
PA2     DAC     n       Word count
        DAC     BUF2    Buffer 2 address
BUF1    BLOK    n       Reserve n words
        DAC     PA2     Automatic reinitialization address for TAR
BUF2    BLOK    n       Reserve n words
        DAC     PA1     Automatic Reinitialization address for TAR
```

NOTE

This example illustrates an application in which the processing rate is slower than the input rate via the channel.

4-6.3.2 UBC Channel Applications

The following examples illustrate four different UBC applications:

Example 1: Simple, single buffer input.

```
        TOA     PA          Parameter Address
        OAW     CU          Initialize PAR and TAR
        TMA     CW          Command Word
        OCW     CU          Initiate Transfer
        BNA     *-1         Delay if channel busy
        . . .
CW      DATA                B23 and others as required by the I/O device.
PA      DAC     n           Word Count
        DAC     BUFF        Address of Input Buffer
BUFF    BLOK    n
```

Example 2: Use Skip Count to read a single word from within a record.

```
        TOA     PA          Parameter Address
        OAW     CU          Initialize PAR or TAR
        TMA     CW          Command Word
        OCW     CU          Initiate Transfer
        BNZ     *-1         Delay if channel is busy
        . . .
CW      DATA                B23 and others as required by the I/O device
        FORM    8,16
PA      DATA    /111,1112/  Word count. Input 112 words from device,
                            skipping first 111.
        DAC     BUFF        Address of Input Buffer
BUFF    BLOK    1           Input Buffer
```

Example 3: Use Automatic Restart to Read a single record into discontiguous buffers.

```
        TOA     PA          Parameter Address
        OAW     CU          Initialize PAR and TAR
        TMA     CW          Command Word
        OCW     CU          Initiate Transfer
        BNZ     *-1         Delay if channel is busy
        . . .
```

```
CW      DATA                B23 and others are required by the I/O device
PA      DAC     n           Word count of input into first buffer
        DAC*    BUF1        Address of first buffer (*) = ARF
        DAC     m           Word count of input into second buffer
        DAC     BUF2        Address of second buffer
        . . .
BUF1    BLOK    n           Reserve n words
BUF2    BLOK    m           Reserve m words
```

Example 4: Use Command Chaining to read two records into a single buffer on some UBC transfer.

```
        TOA     PA          Parameter Address
        OAW     CU          Initialize PAR and TAR
        TMA     CW          Command Word
        OCW     CU          Initiate Transfer
        BNZ     *-1         Delay if channel is busy
        . . .
CW      DATA                B23 and others as required by device to read
                            first record
        . . .
PA      DAC     n           Word count of first record
        DAC*    BUFF,J      Address of buffer for first record. (*) = ARF,
                            Also (,J) = B22 for command and restart
                            B23 and others as required by I/O device to
                            read second record
        DAC     m           Word count of second record
        DAC     BUFF+n      Address of buffer for second record
BUFF    BLOK    n+m         Reserve n+m words
```

4-6.3.3 XBC Channel Applications

The following example illustrates an XBC application:

```
        TOA     INPAD       Set-up Input Buffer Address Start
        OAW     CU          Output the Address to Channel/Unit
        BNZ     *-1         Delay if Channel busy
        TOA     OUTAD       Set-up Output Buffer Address Start
        OAW     CU          Output the Address to Channel/Unit
```

```
        BNZ     *-1             Delay if Channel busy
        TMA     WC      Only    Set-up the required Word Count
        ODW     CU      if      Output the WC to Channel/Unit
        BNZ     *-1     requir. Delay if Channel busy
```

```
        INPAD   BLOK    100     Is the Starting Address of the Input Buffer
                                that device may load data into.*
        OUTAD   BLOK    100     Is the Output Buffer that the device may read
                                data from.*
        WC      DATA    100     Number of Words to Transfer
```

* The external device controls the addressing and interrupt requests to the XBC channel. The external device also controls the word count.

4-7 COMPUTER LINK

By means of this option, two SLASH 6 computers can be connected together to facilitate the transfer of information between them. Distance between the computers may be up to 200 feet. Connection between the computers is via a pair of special cables which transfer control and information signals between Universal Block Controller (UBC) channel boards. Cable connections are made from a UBC channel board in one CPU to a UBC channel board in the second CPU. When interconnected, control signals from one computer can assume control of the UBC channel in the other computer.

In addition to the UBC channel boards, a Programmed I/O Channel (PIOC) board must be installed in each CPU. Up to four external interrupt lines from each of the interrupt generators located on the PIOCs are routed to the priority interrupt circuits of the companion computer. A more detailed explanation of the interrupt generator may be found in Paragraph 2-13.

4-8 MULTI-CPU CHANNEL ADAPTER

4-8.1 General Description

This adapter option enables computer sharing of one or more peripheral devices in a multiprocessor installation. It may also be used in a single computer installation to connect peripheral devices to a channel in a serial-parallel combination rather than the usual serial connection. The adapter is used in conjunction with two types of I/O channel boards; the

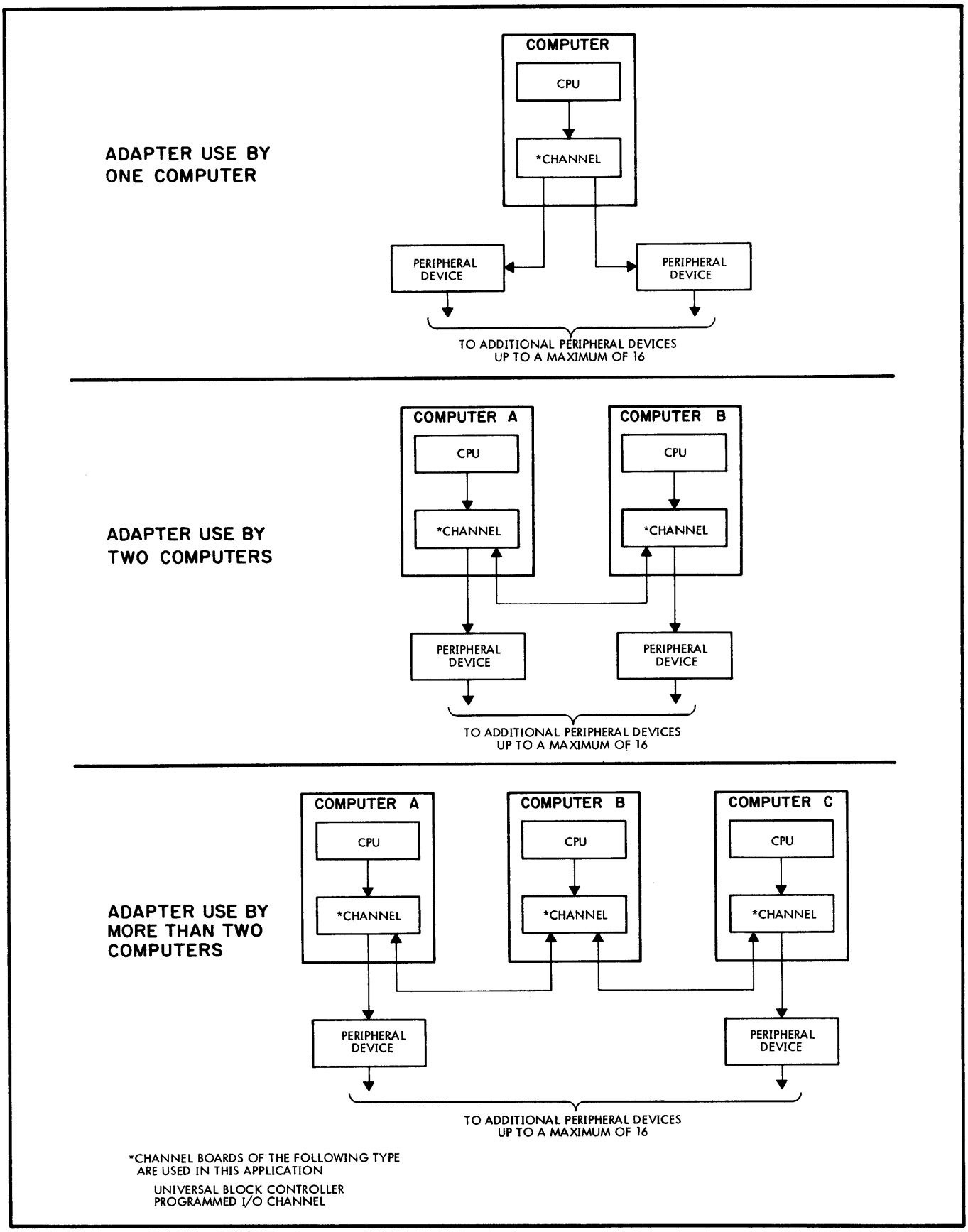
Universal Block Controller (UBC) and the Programmed I/O Channel (PIOC). A set of cables interconnects channel boards when the channel adapter is used.

4-8.2 Operational Description

When the channel adapters are used in a dual computer installation (See Figure 4-8), either computer can access all the peripheral devices assigned to the two channels shown. The CPU in computer A can, for example, generate a channel address for its own channel and a unit address for any peripheral device assigned to its channel or to the channel installed in computer B. In a like manner, computer B can access any unit assigned to the two channels. The total number of peripheral devices that may be connected to the two channels is 16.

Use of the channel adapter option by three computers is shown in Figure 4-8. As in the case of the dual computer configuration, the CPU in computer A may address its own channel and any device connected to its channel or the channel in computer C. Similarly, computers B and C may access any device in the system via their own channels. Because of physical limitations, no peripheral devices are assigned to the channel in computer B. In any multiprocessor installation, peripheral devices can be assigned to only two computers when using the channel adapters. A maximum of 16 devices can be used in configurations of this type.

Normally, peripheral devices are connected (daisy chained) in series to the assigned channel. By use of the channel adapter in a single computer installation, devices may be connected to the channel adapter in a single computer installation, devices may be connected to the channel in a series-parallel combination for certain applications. Two parallel branches of series-connected devices may be tied to the channel with up to a maximum of 16 devices for both branches. Channel and unit addressing is performed as in the standard channel-to-unit configuration.



BD1642-976

Figure 4-8. Multi-CPU Channel Adapter Configurations

SECTION V
PRIORITY INTERRUPT SYSTEM

5-1 GENERAL DESCRIPTION

The SLASH 6 priority interrupt system provides added control over internal CPU operations and I/O functions, and immediate recognition of special external conditions on the basis of predetermined priority. Receipt and recognition of internal or external triggers allows the normal program flow to be diverted to interrupt service subroutines.

Two separate interrupt groups (0 and 1) comprise the SLASH 6 System. Group 0 is reserved for internal CPU functions and is composed of up to eight executive trap levels. Group 1 is reserved for external interrupts. A maximum of 24 external interrupts are available.

5-2 INTERRUPT ORGANIZATION

5-2.1 Priority Conventions

All interrupt levels (both executive traps and external interrupts) are assigned a unique priority number. This assigned priority determines the order in which interrupts will be recognized and serviced. Interrupt levels ascend in order of priority from Group 0, Level 0, to Group 1, Level 23. Group 0 has priority over Group 1; Level 0 has priority over Level 23.

5-2.2 Executive Traps (Group 0)

Each executive trap level is associated with a specific computer feature and is, therefore, permanently assigned. These feature are supplied as options. Each option includes the associated executive trap level(s). Interrupt level assignments for the executive traps (Group 0) are listed below:

0	Power Down	} Power Failure Shutdown/Restart
1	Power Up	
2	Program Restrict	
3	Instruction Trap	
	Or	
3	Instruction Trap	
4	Stall Alarm	
5	Interval Timer	
6	SAU Overflow/Underflow	
7	Address Trap	

5-2.3 External Interrupts (Group 1)

A standard SLASH 6 computer includes interrupt logic and eight individual external interrupt levels. These eight levels are located on the mainframe board and represent Group 1, Levels 16 through 23. Additional external options are available in two groups of eight to provide a maximum of 24 external interrupts. These optional priority interrupt levels are installed on the options board and represent Group 1, Levels 0 through 15. Priority assignments of the interrupt levels are determined by system requirements and are made to meet user's requirements.

5-2.4 Dedicated Memory Locations

Each interrupt level has a memory location dedicated for its exclusive use. This applies to both the executive traps (Group 0) and external interrupts (Group 1). Dedicated memory locations for the interrupt system are described below.

Addresses (Octal)	Assignments (Respective)
60-67	Executive Traps, Levels 0-7
70-117	Group 1 Interrupts, Levels 0-23

5-3 OPERATION AND CONTROL

5-3.1 Basic Operation

Figure 5-1 is a functional block diagram of the SLASH 6 priority interrupt system. Both the executive traps and external interrupts are initiated by a trigger from their assigned functions. The primary operational difference between the two interrupt types is the method of control; executive traps are hardwired in an armed and enabled state, while external interrupts must be previously armed and enabled under program control before an interrupt trigger can be recognized and processed.

5-3.2 Executive Traps (Group 0)

Each executive trap interrupt is designed so as to become active immediately upon receipt of its associated internal trigger, provided no higher-priority level is active. Executive trap interrupt levels are physically integrated with their associated CPU functions (or options), so that installation of the interrupt level is performed simultaneously with installation of the functional logic. Since executive traps are constantly armed and enabled, no program control over the activation of these interrupts is provided.

5-3.3 External Interrupts (Group 1)

External interrupts are program-controlled and not permanently assigned. Program control is afforded by several instructions. Individual levels can be selectively (unitarily) armed, disarmed, enabled, or inhibited under program control. The entire group of interrupts can be simultaneously controlled. For a detailed description of all priority interrupt instructions, refer to the appropriate portion of Section VII in this manual.

Four registers are associated with the external interrupt group. These registers may each be from one to 24 bits wide, depending on the number of interrupt levels within the group. As interrupt levels are added to the system, bits are added to each of the four registers in the group. The register bit positions correspond to the priority level assignments, i.e., bit 0 represents level 0, bit 1 represents level 1, etc. Control of the interrupt registers is accomplished by the following group of instructions.

1. Transfer Double to group 1 (TD1)
2. Transfer group 1 to Double (T1D)
3. Transfer Double to group 1 (TD4 - software - triggered interrupt)
4. Transfer group 1 to Double (T4D - software interrupt status)

The armed/disarmed and enabled/inhibited states of each interrupt level are retained in the Arm/Disarm (A/D) and Enable/Inhibit (E/I) registers, respectively. A TD1 instruction is used to selectively arm, disarm, enable, or inhibit individual interrupt levels within the group. Upon execution of a TD1 instruction, the contents of the E and A registers are transferred, respectively, to the A/D and E/I registers in group 1. Transfers are performed in a bit-for-bit pattern. A ONE is a given bit position of the A/D register will cause the corresponding interrupt level to be armed; a ZERO will disarm the level. An interrupt will be enabled or inhibited by a ONE or ZERO, respectively, in the corresponding bit position of the E/I register.

The interrupt group's armed/disarmed and enabled/inhibited status may be determined under program control by the execution of a T1D instruction. The contents of the A/D and E/I registers are transferred to the E and A registers, respectively. A/D and E/I register contents are not affected by the transfer.

External interrupt triggers normally occur asynchronously with respect to CPU operation. However, interrupt triggers can be generated under program control by a TD4 instruction. The TD4 instruction performs a logical OR between the contents of the E and A registers and the interrupt Request and Active registers, respectively. Loading the Request register with a ONE has the same effect as an external trigger at the corresponding interrupt level. When the Active register is loaded with a ONE, the corresponding level will become active as long as no higher-level interrupt

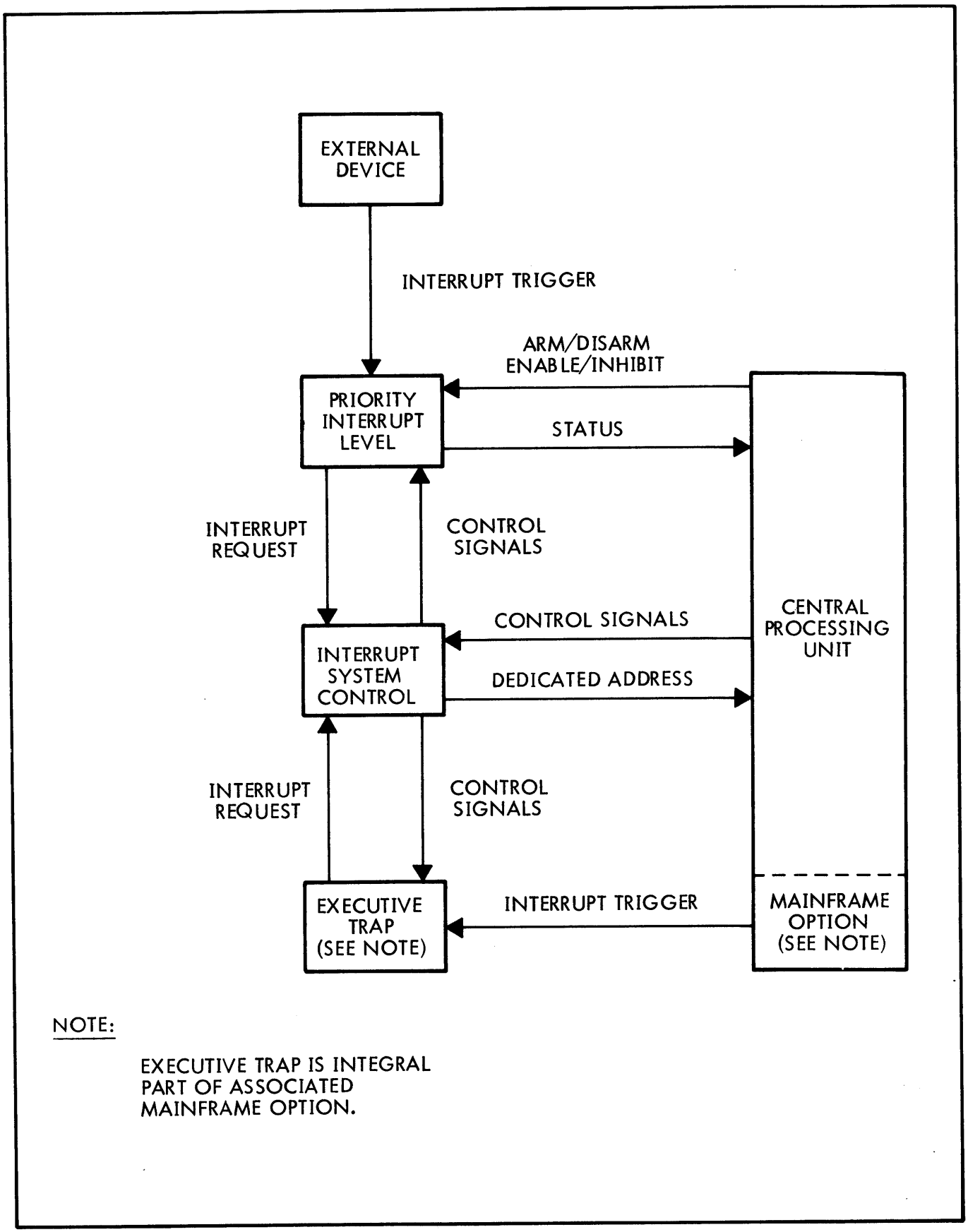


Figure 5-1. Functional Block Diagram, Priority Interrupt System

is active. The T4D instruction transfers the Request and Active register's contents to the E and A registers, respectively. The Request and Active registers are not affected.

Figure 5-2 illustrates the control system for external interrupts. Each external interrupt operates in three distinct states: inactive, waiting, and active. In the inactive state, the level has not received an interrupt trigger. When a trigger is received, the armed/disarmed status determines whether the triggered interrupt will be placed in a waiting state or ignored. If the triggered interrupt is armed, it will be placed in the waiting state, if disarmed, it will be ignored.

If an interrupt is armed but inhibited (i.e., not enabled), it is held in the waiting state until such time as it is enabled under program control. Once enabled, the interrupt will become active as soon as the current instruction is completed, assuming that no higher level is active and that external interrupts are not being held (HXI instruction).

Once an interrupt becomes active, it can be inhibited under program control (TD1 instruction). This places the active level in an off-line mode or permissive state. The permissive state does not affect execution of the interrupt subroutine but enables lower priority armed and enabled interrupts to become active when triggered. For example, if active level two is inhibited by the program, waiting level three becomes active immediately. After level three is serviced, the processing of the level-two subroutine is resumed until it is completed or another interrupt becomes active. Should another interrupt trigger be received by an interrupt that is in the permissive state, it will be saved and recognized when that level is returned to the on-line mode.

Hold and Release eXternal Interrupts (HXI and RXI) instructions are employed to prohibit and restore the activation of any external interrupt (other than currently-active levels) regardless of that interrupt's armed/disarmed and enabled/inhibited states. Such a prohibition would ensure that another, lower-level, interrupt could complete its processing routine without interruption. This hold condition can only be released by an RXI instruction.

Several instructions are privileged. Should an interrupt occur during the execution of one of these instructions, it will not be allowed to become active until the completion of the instruction following the privileged instruction. The privileged instructions are:

1. Branch and Save return - Long (BSL)
2. Hold interrupts and Transfer I to memory (HTI)
3. Hold interrupts and Transfer J to memory (HTJ)
4. Hold interrupts and transfer K to memory (HTK)
5. Release eXternal Interrupts (RXI)
6. EXecute Memory (EXM)

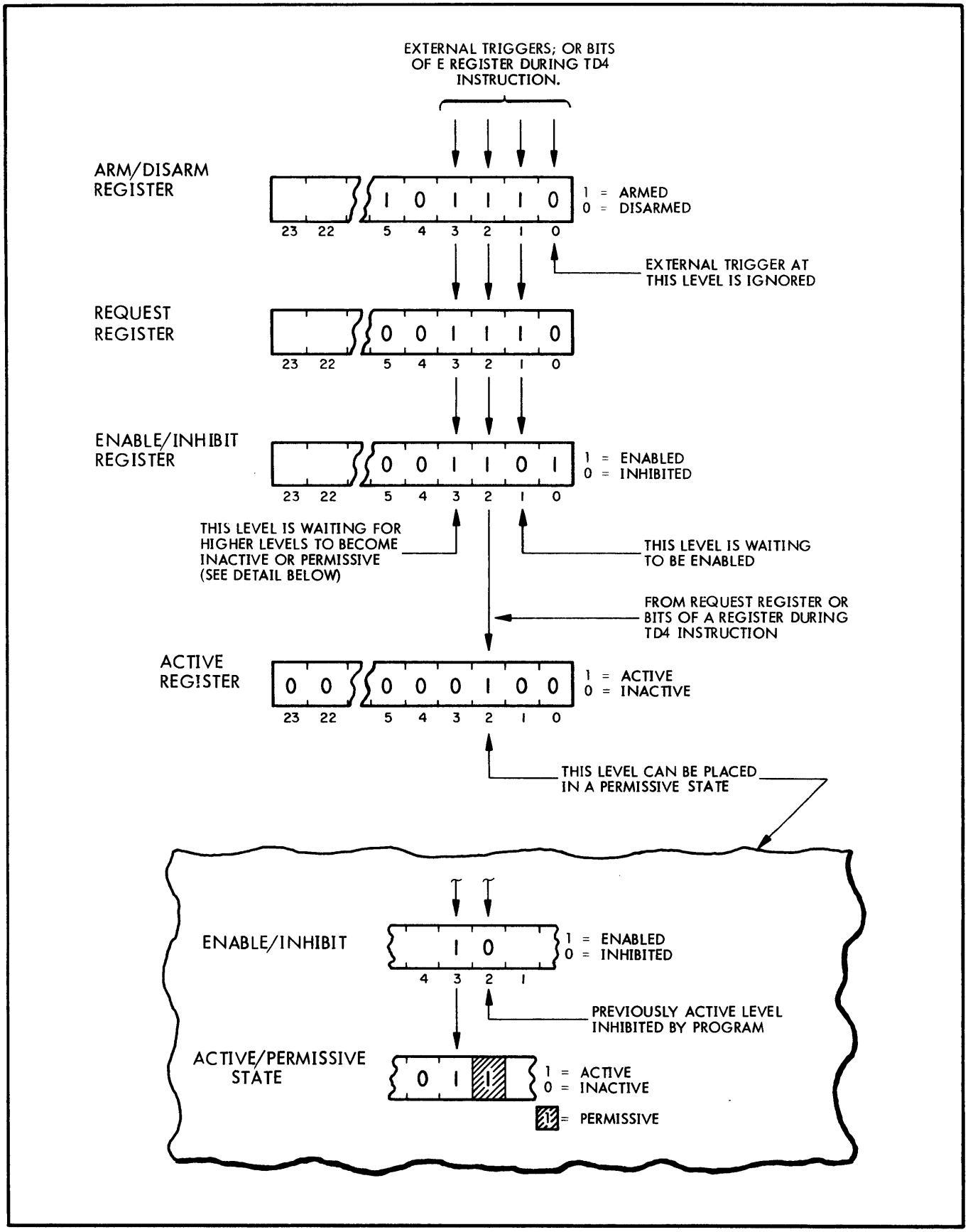
7. Transfer Memory to Registers (TMR)
8. Transfer Registers to Memory (TRM)
9. Update Stack Pointer (USP)
10. Transfer Double to group 1 (TD1)
11. Transfer Double to group 1 (TD4)
12. Unitarily Arm group 1 interrupts (UA1)
13. Unitarily Disarm group 1 interrupts (UD1)
14. Unitarily Enable group 1 interrupts (UE1)
15. Unitarily Inhibit group 1 interrupts (UI1)

5-4 INTERRUPT PROCESSING CONSIDERATIONS

Each external interrupt and executive trap level is assigned a unique memory location (reference: Paragraph 5-2.4). An interrupts, when activated, generates an address and an instruction operation code. The address specifies the dedicated location and the operation codes define an eXecute Memory (EXM) instruction. The address and EXM instruction are placed in the Instruction Register, decoded, and executed as a normal operation. This causes the instruction in the dedicated location to be executed as if it were the next instruction in the main program.

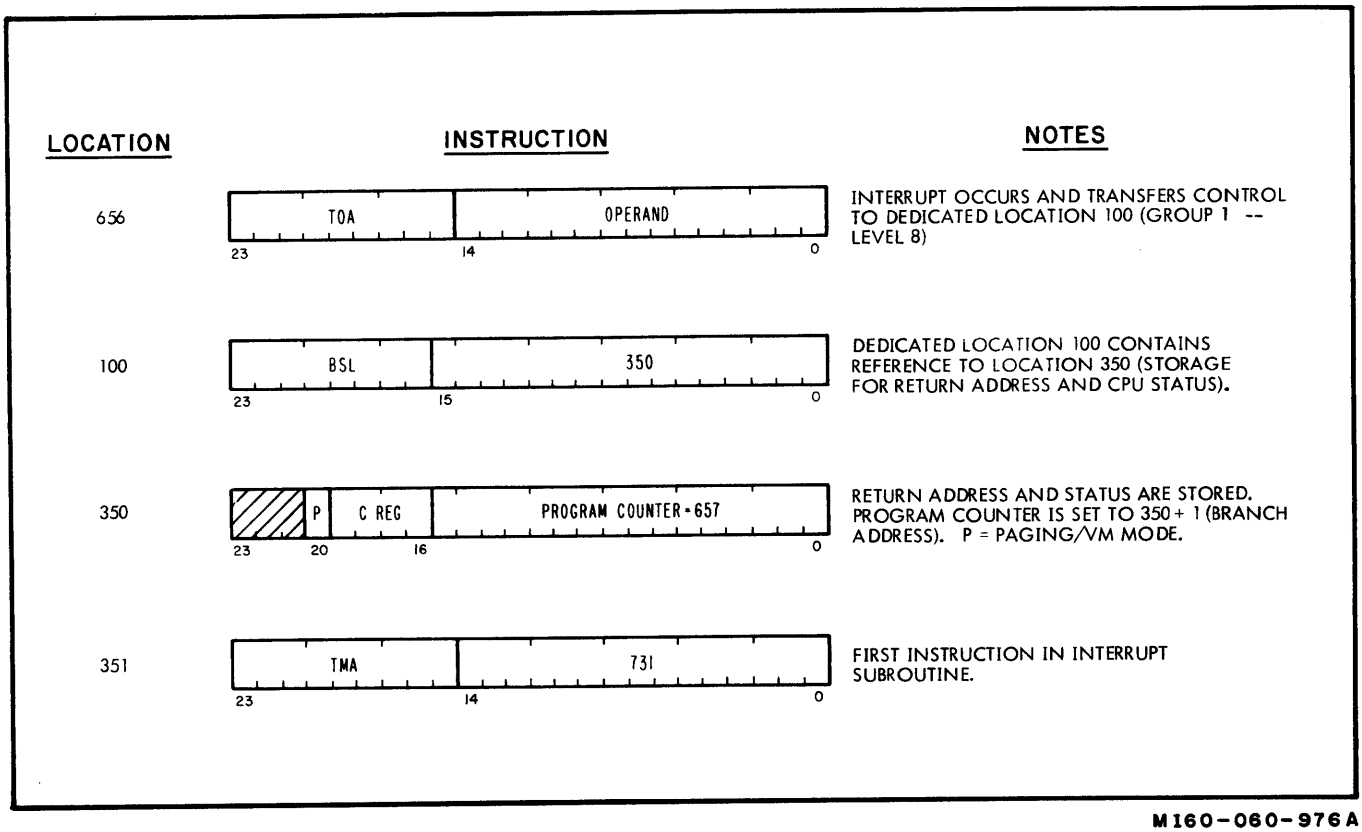
Although any instruction may be stored in an interrupt's dedicated memory location, the operation designed for subroutine entry is the Branch and Save return - Long (BSL) instruction. The BSL instruction is used to enter an interrupt subroutine because it provides a means of saving machine status and returning to the program location following that being executed at the time of the interrupt. When an interrupt is generated, the current instruction is allowed to continue so the program counter can be advanced before interrupt processing begins. Figure 5-3 illustrates the sequence of events.

A means of exit from the interrupt routine is the Branch and Reset interrupt - Long (BRL) instruction. Normally, the BRL instruction would make use of an indirect reference (*) to the address previously referenced by the BSL instruction upon entering the routine. If this is done, the Condition register is restored to its original contents (at the time the interrupt occurred). Figure 5-4 illustrates the subroutine exit sequence.



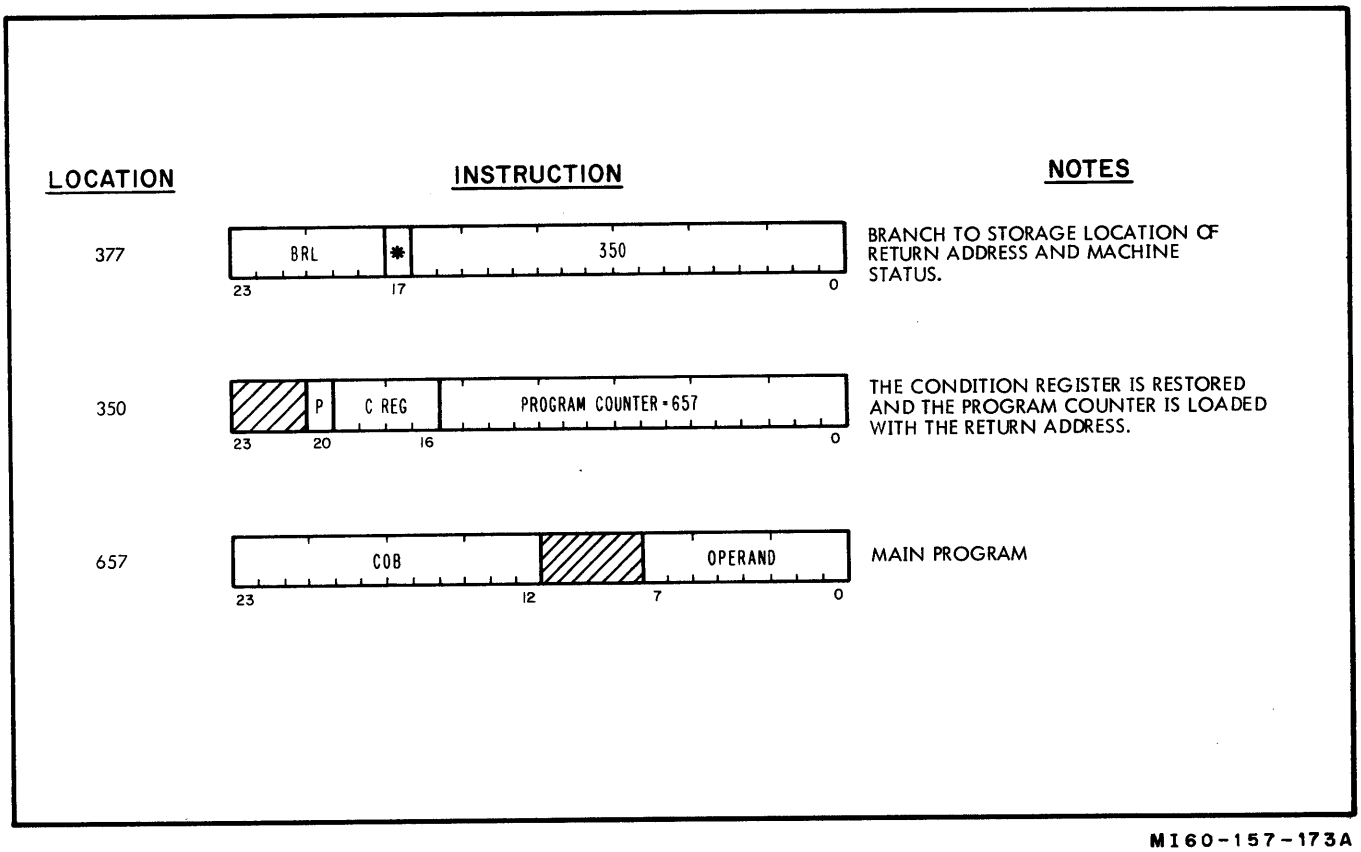
MI60-100-976B

Figure 5-2. External Interrupt Control



M160-060-976A

Figure 5-3. Interrupt Subroutine Entry



M160-157-173A

Figure 5-4. Interrupt Subroutine Exit

The BRL instruction resets the highest active (not in permissive state) trap or external interrupt level provided that external interrupts are not being "held" (HXI instruction). Active traps are always reset by the BRL instruction. Active interrupts can only be reset by the BRL instruction, a TD1 instruction, or by master clearing the CPU.

NOTE

A BRL instruction will not reset an interrupt that is in the permissive state.

SECTION VI
SCIENTIFIC ARITHMETIC UNIT (SAU)

6-1 GENERAL DESCRIPTION

The optional Scientific Arithmetic Unit (SAU) provides concurrent double-precision, floating-point capability for the computer. When used with the SLASH 6 digital computer, the SAU implements the execution of 47 additional instructions or operation codes. Of these instructions, 19 permit concurrent computer/SAU operations. SAU data and condition information are displayed on the Programmers Control Panel as a function of selectable shared indicators.

6-2 FLOATING-POINT DATA FORMATS

All arithmetic operations are carried out in double-precision format to yeidl a 39-bit mantissa and an 8-bit exponent. Figure 6-1 illustrates the floating-point data formats employed by the CPU's Double (D) register, memory, and the SAU's X and XW registers.

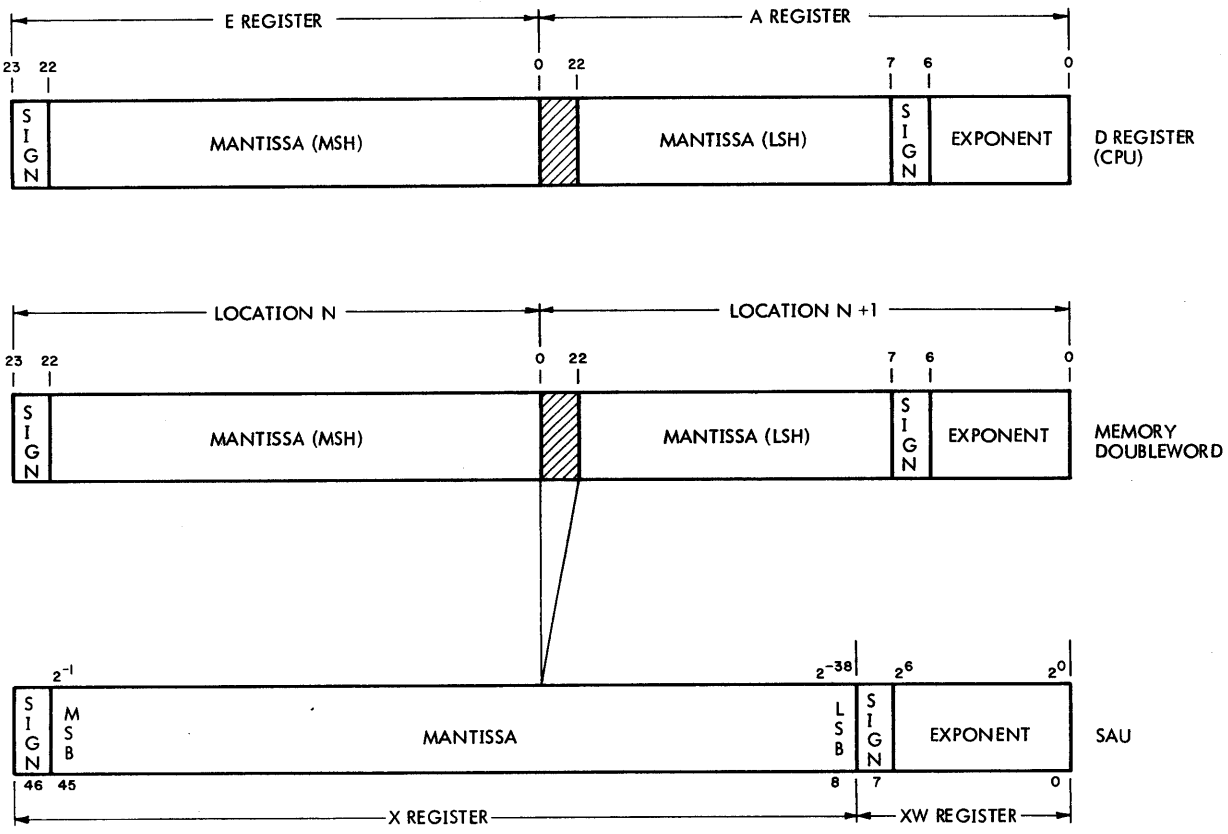
Data transfers to the SAU from the CPU are either single-precision integers or double-precision, floating-point, normalized numbers. All arithmetic operations performed within the SAU are executed in the double-precision, floating-point format illustrated in Figure 6-1. Therefore, any integer number transferred to the SAU for arithmetic operations is first normalized and converted to floating-point format within the SAU. All double-precision transfers to the SAU, whether from the D register or memory, are assumed to be normalized, floating-point quantities. Bit 23 of the least-significant half (LSH) of the double word is truncated.

6-3 SAU REGISTERS

Three SAU registers are available to the programmer. These are:

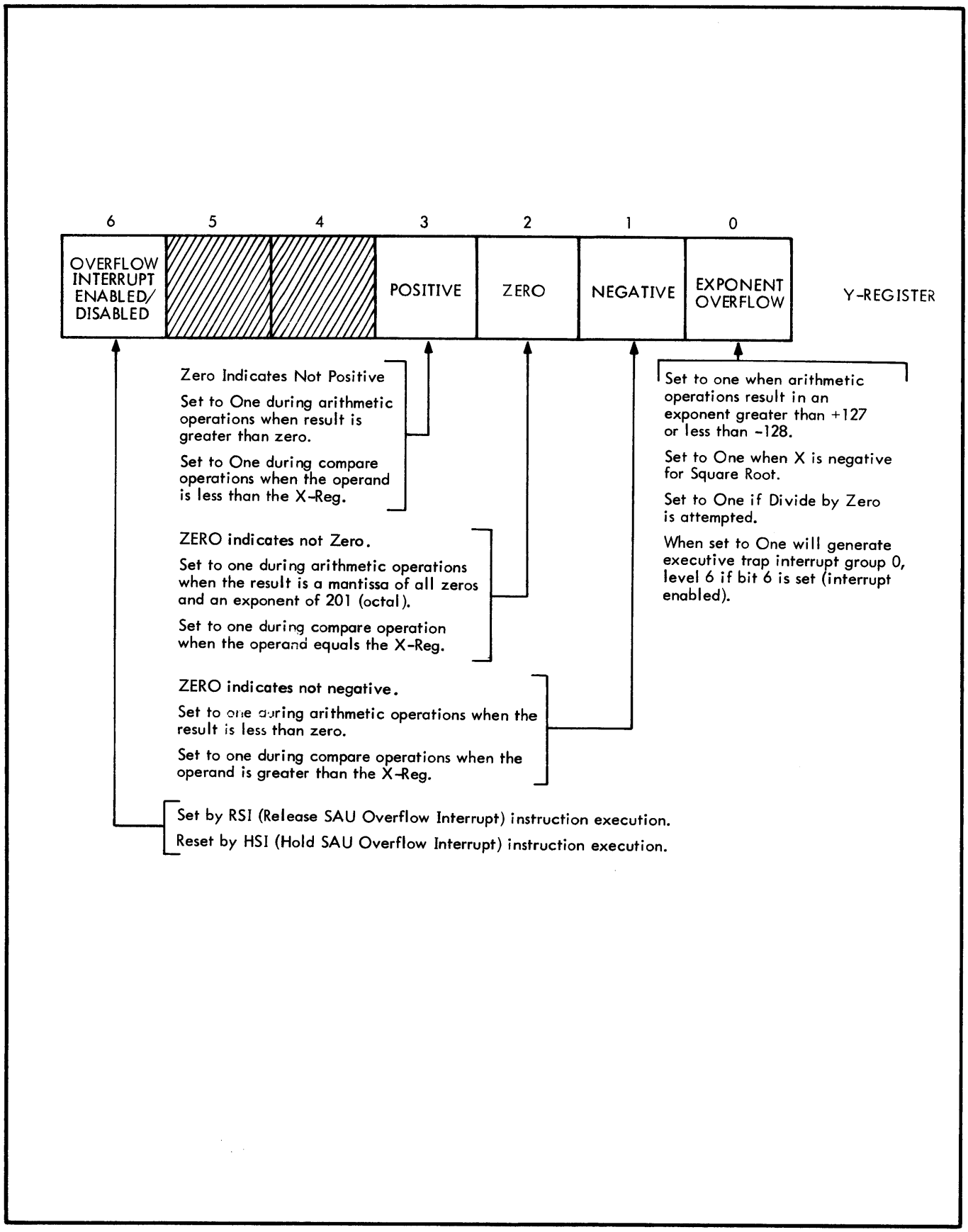
a. X register (signed mantissa - Figure 6-1);
b. XW register (signed exponent - Figure 6-1); and
c. Y register (SAU condition - Figure 6-2).

The XW register can be independently modified via the SAU instruction set. Figure 6-2 illustrates the Y (condition) register bit configuration and their significance in reflecting the results of SAU operations.



MI1226-976B

Figure 6-1. Floating-Point Data Formats



MI1227 - 673

Figure 6-2. SAU Y (Condition) Register

6-4 OPERATION AND CONTROL

6-4.1 Data Transfers

A simplified block diagram of the SAU in relation to the CPU is shown in Figure 6-3. All data transfers between the CPU and SAU are, effectively, confined to the X, XW, and Y registers. CPU-SAU data transfers may involve the E and A registers or memory. The transfer source and destination are selected as a function of the instruction being executed. In all double-precision transfers to an from SAU, the least-significant half (LSH) is transferred first. When memory is involved in the double-precision transfer memory, location N+1 (refer to Figure 6-1) must be addressed before location N in order to maintain the proper format. The CPU controls this addressing sequence as a normal instruction execution function.

6-4.2 SAU Instructions

Table 6-1 lists and defines the instruction set for the SAU. The table shows execution times in terms of CPU cycles and also lists the available concurrent cycles available for processing other (non-SAU) instructions during SAU "busy" periods. For a detailed description of SAU instructions, refer to Section VII of this manual.

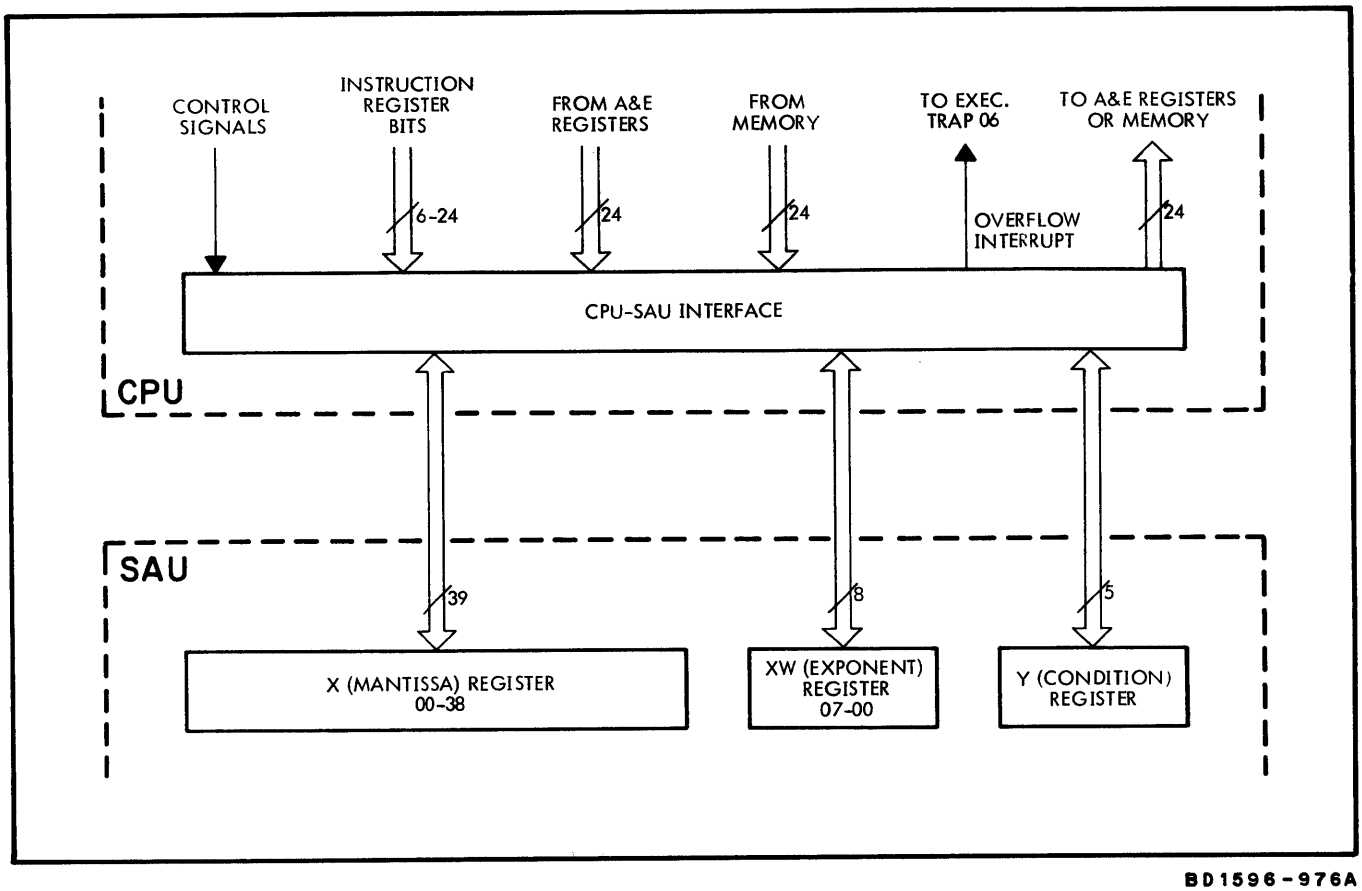


Figure 6-3. CPU-SAU Transfer Paths; Simplified Block Diagram

Table 6-1. SAU Instruction Set

Mnemonic	Description	Mainframe Cycles	Concurrent Cycles Available
TMX	Transfer Memory to X		
TXM	Transfer X to Memory		
AMX	Add Memory to X		
SMX	Subtract Memory from X		
MMX	Multiply Memory by X		
DMX	Divide Memory into X		
BOX	Branch on SAU Ready		
BNR	Branch on Negative Reset		
BNS	Branch on Negative Set		
BZR	Branch on Zero Reset		
BZS	Branch on Zero Set		
BPR	Branch on Positive Reset		
BPS	Branch on Positive Set		
BOR	Branch on Overflow Reset		
BOS	Branch on Overflow Set		
TYA	Transfer Y to A		
TOY	Transfer Operand to Y		
TOW	Transfer Operand to W		
AOW	Add Operand to W (Exponent)		
COW	Compare Operand to W (Exponent)		
HSI	Hold SAU Interrupt		
RSI	Release SAU Interrupt		
FAX	Floating Normalize of A to X		
PXX	Positive of X to X		
NXX	Negative of X to X		
TZX	Transfer Zero to X		
INX	Inverse of X to X		
SEX	Square of X to X		
SRX	Square Root of X to X		
CZX	Compare Zero to X		
AOX	Add Operand and X		
SOX	Subtract Operand from X		
MOX	Multiply Operand and X		
DOX	Divide Operand into X		
AAX	Add A and X		
SAX	Subtract A from X		
MAX	Multiply A and X		
DAX	Divide A into X		
ADX	Add D and X		
SDX	Subtract D from X		
MDX	Multiply D and X		
DDX	Divide D into X		
IDX	Interchange D and X		
CDX	Compare D and X		
FXA	Fix of X to A		
TDX	Transfer D to X		
TXD	Transfer X to D		

6-5 PROGRAMMING CONSIDERATIONS

The SAU and CPU will operate concurrently for one or more machine cycles, depending on the SAU instruction being executed. In order to take advantage of the available cycles, CPU and SAU instructions must be intermixed.

If the instruction sequence contains several consecutive SAU instructions, the CPU will wait for the SAU, i.e., if an SAU instruction is in progress and another SAU instruction follows it, the CPU must wait until the second instruction has started (or completed, if there is no time-sharing) before executing any non-SAU instruction. For example the sequence:

TMX	A	(3 cycles)
MMX	B	(7 cycles; 3 to initiate, 4 for concurrent operation)
DMX	C	(16 cycles; 3 to initiate, 13 for concurrent operation)
TXM	D	(3 cycles)

will not permit execution of non-SAU instructions for 29 cycles. Note, however, that there are 17 cycles available in the sequence for execution non-SAU instructions. The following sequence makes use of the available CPU cycles.

TMA	A		
MMX	B		
TMD	X	(3 cycles)	4 Concurrent MMX cycles
TOI	30	(1 cycle)	
DMX	C		
AMD	Y	(3 cycles)	
TMD	Z	(3 cycles)	
AMI	J	(2 cycles)	
TIA		(1 cycle)	13 Concurrent DMX cycles
NII		(1 cycle)	
AAM	K	(3 cycles)	
TXM	D		

6-6 SAU INTERRUPT

The executive trap (Group 0, Level 6) provided with the SAU is used to detect overflow/underflow conditions resulting from the execution of SAU instructions. The trap is controlled by two SAU instructions and the Hold/Release external interrupt instructions of the CPU.

The SAU instructions which control the trap are:

HSI	-	Hold SAU overflow interrupt
RSI	-	Release SAU overflow interrupt.

The trap, when enabled, is triggered by the overflow bit (bit 0) of the SAU condition register (Y register). In order to start SAU operation and enable the trap the following sequence may be used.

```
TOY     0     or     TMX          OPERAND
RSI                  RSI
```

Either sequence clears the overflow bit and prevents an extraneous interrupt.

When the SAU trap is enabled and an overflow occurs, the SAU is set to a busy condition, preventing the execution of any other SAU instruction except a HSI. This allows the program to determine the location of the SAU instruction which caused the overflow. The SAU interrupt processing routine must execute an HSI as its first SAU instruction. Prior to exiting the service routine, bit 0 of the Y register must be cleared and an RSI instruction performed to rearm the SAU trap. A typical entry/exit sequence is:

```
SAUPI   ***
        HSI
        .
        .
        .
        TOY     0
        RSI
        BRL*    SAUPI
```

Note that an overflow can be caused by program control with the sequence:

```
        HSI
        RSI
        TOY     1
```

It should be noted that the contents of the Program Counter at the time of the interrupt does not necessarily have a direct relation to the location of the SAU instruction which caused the overflow. This is due to the concurrent processing capability, the occurrence of other interrupts, the execution of the HXI/RXI instructions and the way in which the SAU and CPU instructions are intermixed.

When it is a requirement to know exactly where the instruction causing the overflow is located, careful coding ismandatory if the concurrent operation capability is to be used. It is recommended that, in cases where overflow is likely, the SAU instructions be written consecutively to simplify the procedure for finding which SAU instruction caused the overflow.

SECTION VII
INSTRUCTION SET

7-1 INTRODUCTION

The SLASH 6 Computer System instruction set consists of several functional groups or families of instructions. Among these are: Arithmetic; Branch; Compare; Input/Output; Logical; Shift; Transfer; etc. Each group, in turn, is composed of individual instructions that perform specific functions.

Through the application of the instruction set, the programmer has access to each memory location and major register in the CPU. In addition, the instruction set provides for the alteration and control of program flow, manipulation and modification (arithmetic and logical) of data, servicing of priority interrupts and control of I/O operations.

7-2 INSTRUCTION FORMATS

Each instruction is decoded from a 24-bit memory word. The instruction word bits define the operation to be performed and the manner in which it is to be performed. All instruction formats contain an operation code (Op Code) that defines the general process that is to be undertaken (Add, Subtract, Interchange, etc.). The Op Code usually contains either six or 12 bits; a few instructions require expansion of the Op Code beyond 12 bits.

Additional bits in the instruction word specify how the general operation is to be performed. For example, when adding the contents of one register to the contents of another, the additional bits indicate which registers are involved.

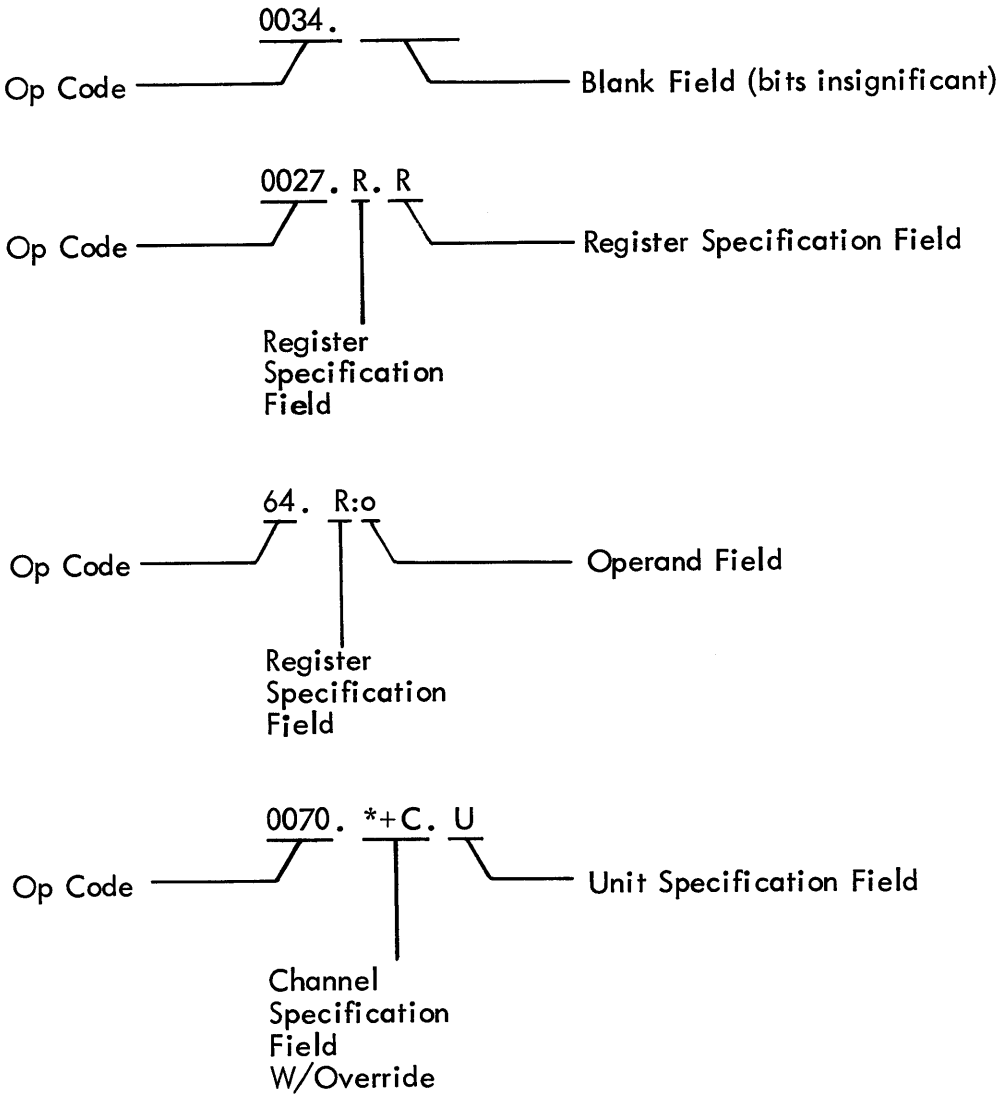
Some instructions access memory and use formats that specify an address. The address bits are sometimes supplemented by special bits (indirect, index) in the instruction word. In other cases, the additional bits are not used for address modification, but are used to define a condition under which the specified memory location will be accessed or to indicate which of the CPU registers will be used in the operation. The appropriate formats are provided with the individual instruction descriptions.

7-3 INSTRUCTION FORMULA

The instruction formula, presented with each instruction description, provides a graphic representation of a 24-bit instruction word. The formula expresses an instruction word as a concatenation of its various fields where each field is represented by one or more octal digits.

For example, the formula 21.*+X:a expresses a memory reference branch where: 21 represents a 6-bit (2 octal digits) Op Code, * and X are additive quantities defining the indirect (*) and index (X) field, and "a" is a memory reference in a 15-bit address field.

The period (.) and colon (:) provide field separation in the formula, with the colon indicating right/left justification. All digits or references to the left of the colon are left-justified, and those to the right are right-justified in their respective fields. The absence of a colon indicates that all digits or references are left-justified in their fields. Examples of instruction formulas are as follows:



7-4 INSTRUCTION DESCRIPTIONS

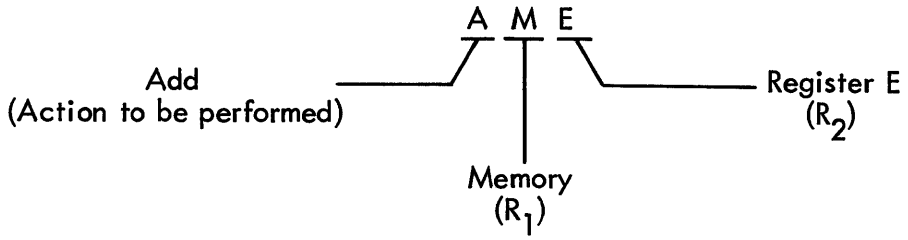
The following paragraphs describe, in detail, the various instructions in the SLASH 6 repertoire. The instructions are arranged by functional groups (Arithmetic, Branch, Compare, etc.). General information pertaining to each group is presented in the introductory paragraphs.

Each instruction description includes the three-letter mnemonic identifier, instruction name, instruction formula, and lists the registers affected. Bit assignments for each instruction are shown by means of the binary word format illustration. The number of cycles required to perform the instruction and a brief-explanation of the instruction operation is provided. Special notes are given, where required, to complete the instruction description.

7-5 ARITHMETIC INSTRUCTIONS

The SLASH 6 Arithmetic instruction group includes the standard arithmetic operations –addition, subtraction, multiplication and division – as well as square root, normalization and sign extension instructions. Also included are several register-to-register operations which compute the absolute value, negate or round off the contents, or negate the sign of one register and subsequently transfer its contents to a second register.

The arithmetic instruction mnemonics provide a brief definition of specific operations to be performed. The first letter of the mnemonic signifies the action or type of operation to be performed, the second letter identifies the first quantity or reference (R_1) to be used in the operation, and the third letter identifies the second reference (R_2). For example:



In the majority of arithmetic instructions, the result of the operation remains in R_2 leaving R_1 unchanged (except where R_1 and R_2 are the same). Certain instructions - notably, those performing multiplication, division, sign extension and square root computation - do not comply with the R_1/R_2 conventions stated above. These instructions are described thoroughly in the individual instruction descriptions.

Unless noted otherwise, each arithmetic operation causes the Condition (C) register to be set reflecting the status of the result. The various arithmetic conditions are defined below:

a. Positive - Result is arithmetically greater than zero, indicated by a ONE in bit position 3 of the C register. A ZERO in bit position 3 indicates " Not Positive" .

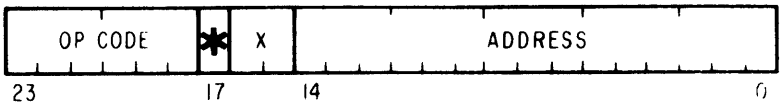
b. Zero - All bits of the quantity under consideration are ZEROs, indicated by a ONE in bit position 2 of the C register. A ZERO in bit position 2 indicates " Not Zero" .

c. Negative - Result is arithmetically less than zero, indicated by a ONE in bit position 1 of the C register. A ZERO in bit position 1 indicates " Not Negative" .

d. Overflow - An Overflow results from an operation instead of displaying the status of an operand. As a general rule, an arithmetic Overflow will occur when a bit is carried into the designated sign bit position and not carried out or vice versa. An Overflow condition is indicated by a ONE in bit position 0 of the C register. A ZERO in bit position 0 indicates " No Overflow" .

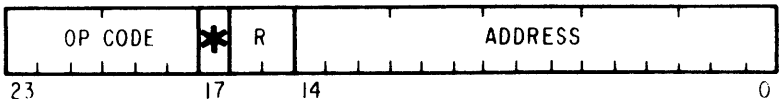
The following instructions are included in the Arithmetic Group.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
30.*+X:a	Add Unity to Memory	M,C	AUM



The contents of the effective memory address are incremented by one.

41.*+1:a	Add Memory to I	I,C	AMI
41.*+2:a	Add Memory to J	J,C	AMJ
41.*+3:a	Add Memory to K	K,C	AMK

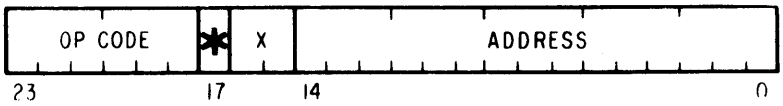


The contents of the effective memory address are algebraically added to the contents of register I, J or K.

NOTE

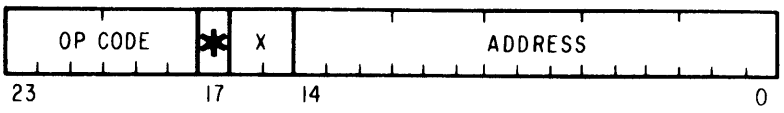
The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted.

42.*+X:a	Add Memory to E	E,C	AME
43.*+X:a	Add Memory to A	A,C	AMA



The contents of the effective memory address are algebraically added to the contents of register E or A.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
44.*+X:a	Add Memory to Double	E,A,C	AMD

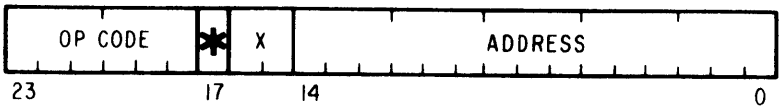


The contents of the effective memory address (EMA) and the next sequential memory address (EMA+1) are algebraically added to the contents of register D according to the double integer format defined in Section II.

NOTE

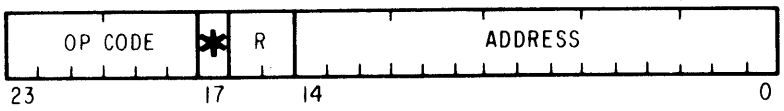
Bit A_{23} must be ZERO.

45.*+X:a	Add Memory to Byte	A,C	AMB



Bits 0-7 of the contents of the effective memory address are algebraically added to the contents of register B (A_0-A_7). Bits 8-23 of register A are unchanged.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
46.*+1:a	Add I to Memory	M,C	AIM
46.*+2:a	Add J to Memory	M,C	AJM
46.*+3:a	Add K to Memory	M,C	AKM



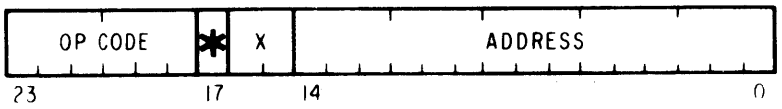
The 24-bit contents of register I, J or K are algebraically added to the contents of the effective memory address.

NOTE

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,

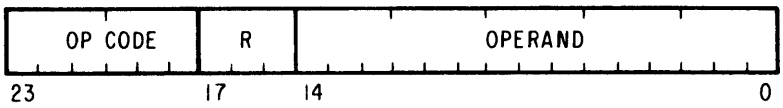
```
      AJM*    X
       .
       .
       .
X     DAC     Y,K
```

47.*+X:a	Add E to Memory	M,C	AEM
50.*+X:a	Add A to Memory	M,C	AAM



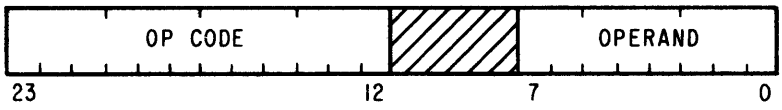
The contents of register E or A are algebraically added to the contents of the effective memory address.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
64.1:o	Add Operand to I	I,C	AOI
64.2:o	Add Operand to J	J,C	AOJ
64.3:o	Add Operand to K	K,C	AOK
64.4:o	Add Operand to E	E,C	AOE
64.5:o	Add Operand to A	A,C	AOA
64.6:o	Add Operand to T	T,C	AOT



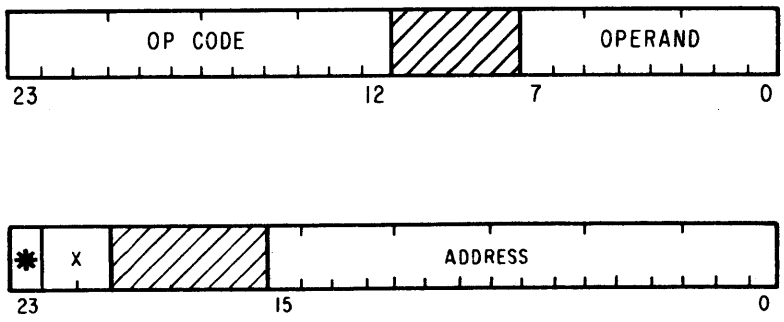
The 15-bit unsigned operand is algebraically added to the contents of the specified register.

0012:o	Add Operand to Byte	A,C	AOB



The 8-bit signed operand is algebraically added to the contents of the B register. (A_0-A_7). Bits 8-23 of register A are unchanged.

INSTRUCTION FORMULA		FUNCTION	REGISTERS AFFECTED	MNEMONIC
0074:o *+X:A	(word 1) (word 2)	Add Operand to Memory	M,C	AOM (n) DAC (m)



The 8-bit signed operand (n) is algebraically added to the contents of the effective memory address (m).

NOTES

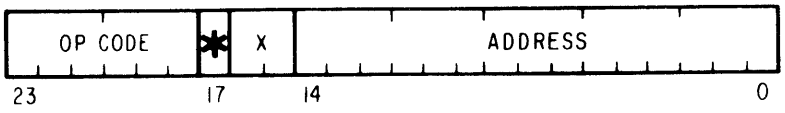
1. If a demand page, restrict mode violation, or limit violation occurs when attempting to access the effective memory address while in the virtual memory User mode, the Program Counter will automatically be decremented by two. If the violation occurs during the fetch of the second word, the Program Counter will be decremented by one.

2. An AOM instruction may not be used after a ROM instruction (Reference: Paragraph 7-13).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0020.01.01	Add I to I	I,C	AII
0020.01.02	Add I to J	J,C	AIJ
0020.01.04	Add I to K	K,C	AIK
0020.01.10	Add I to E	E,C	AIE
0020.01.20	Add I to A	A,C	AIA
0020.01.40	Add I to T	T,C	AIT
0020.02.01	Add J to I	I,C	AJI
0020.02.02	Add J to J	J,C	AJJ
0020.02.04	Add J to K	K,C	AJK
0020.02.10	Add J to E	E,C	AJE
0020.02.20	Add J to A	A,C	AJA
0020.02.40	Add J to T	T,C	AJT
0020.04.01	Add K to I	I,C	AKI
0020.04.02	Add K to J	J,C	AKJ
0020.04.04	Add K to K	K,C	AKK
0020.04.10	Add K to E	E,C	AKE
0020.04.20	Add K to A	A,C	AKA
0020.04.40	Add K to T	T,C	AKT
0020.10.01	Add E to I	I,C	AEI
0020.10.02	Add E to J	J,C	AEJ
0020.10.04	Add E to K	K,C	AEK
0020.10.10	Add E to E	E,C	AEE
0020.10.20	Add E to A	A,C	AEA
0020.10.40	Add E to T	T,C	AET
0020.20.01	Add A to I	I,C	AAI
0020.20.02	Add A to J	J,C	AAJ
0020.20.04	Add A to K	K,C	AAK
0020.20.10	Add A to E	E,C	AAE
0020.20.20	Add A to A	A,C	AAA
0020.20.40	Add A to T	T,C	AAT
0020.40.01	Add T to I	I,C	ATI
0020.40.02	Add T to J	J,C	ATJ
0020.40.04	Add T to K	K,C	ATK
0020.40.10	Add T to E	E,C	ATE
0020.40.20	Add T to A	A,C	ATA
0020.40.40	Add T to T	T,C	ATT

The contents of R_1 are algebraically added to the contents of R_2.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
57.*+X:a	DiVide by Memory	E,A,C	DVM



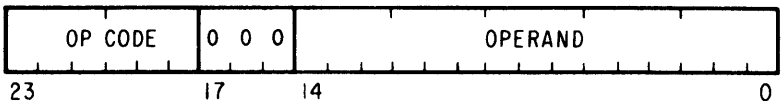
The double-precision contents of register D (E and A) are algebraically divided by the single-precision contents of the effective memory address. The signed, single-precision, quotient is left in A and the remainder is left in E. The remainder will have the same sign as the original dividend and the Condition register will be set according to the status of the quotient.

NOTES

1. If it is desired to divide a single-precision number in A by memory, an Extend Sign of A (ESA) instruction should be executed prior to the DVM. This will establish the proper format for the dividend.

2. If the contents of E are equal to, or greater than, the contents of memory, an Overflow condition will result and the Condition register will be set accordingly.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
61.0:o	DiVide by Operand	E,A,C	DVO



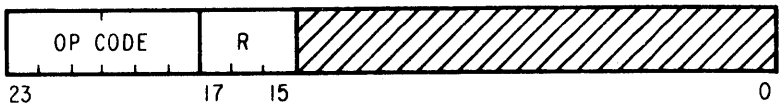
The double-precision contents of register D (E and A) are algebraically divided by the 15-bit unsigned operand. The signed, single-precision, quotient is left in A and the remainder is left in E. The remainder will have the same sign as the original dividend and the Condition register will be set according to the status of the quotient.

NOTES

If it is desired to divide a single-precision number in A by the operand, an Extend Sign of A (ESA) instruction should be executed prior to the DVO. This will establish the proper format for the dividend.

If the contents of E are equal to, or greater than, the operand, an Overflow condition will result and the Condition register will be set accordingly.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
61.1	DiVide by I	E,A,C	DVI
61.2	DiVide by J	E,A,C	DVJ
61.3	DiVide by K	E,A,C	DVK
61.6	DiVide by T	E,A,C	DVT



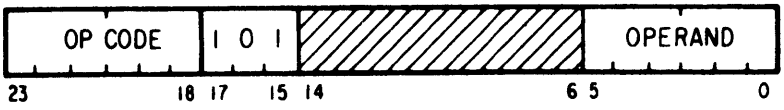
The double-precision contents of register D (E and A) are algebraically divided by the specified register. The signed, single-precision, quotient is left in A and the remainder is left in E. The remainder will have the same sign as the original dividend and the Condition register will set according to the status of the quotient.

NOTES

1. If it is desired to divide a single-precision number in A by the contents of the specified register, an Extend Sign of A (ESA) instruction should be executed prior to the divide instruction. This will establish the proper format for the dividend.

2. If the contents of E are equal to, or greater than, the contents of the specified register, an Overflow condition will result and the Condition register will be set accordingly.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
61.5:0	DiVide by 2	E	DV2

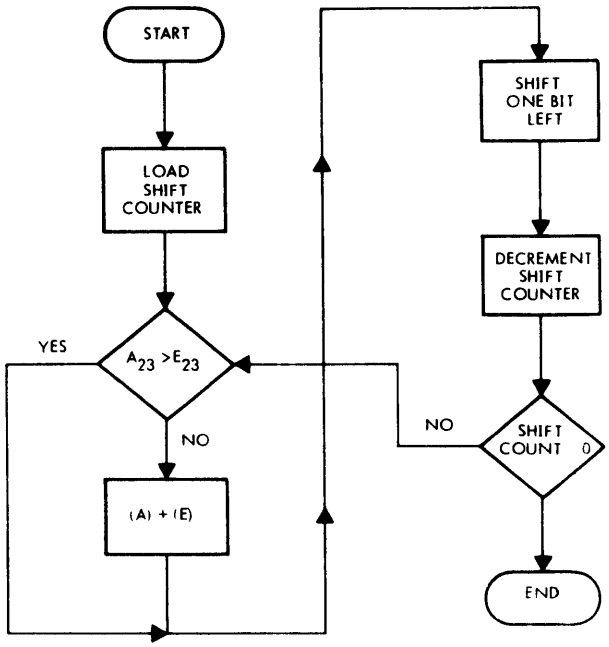


The DV2 instruction divides the contents of the E accumulator by the contents of the A accumulator, except that the arithmetic operation will be Modulo 2 (exclusive OR) instead of 2's complement arithmetic. The 6-bit operand contained in the instruction specifies the number of shifts.

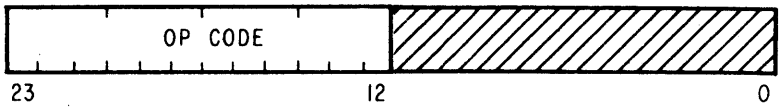
NOTES

The specified number of shifts must be an even number.

This instruction is used for generating and checking error codes based on polynomial coding techniques. The polynomial and the operand to be implemented must be left-justified in the A and E accumulators. The result will be placed in the E accumulator while the polynomial will remain in the A accumulator.

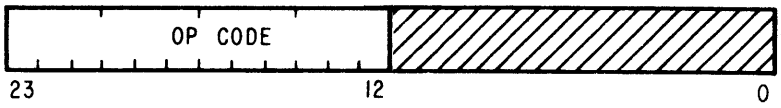


INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0037.	Extend Sign of A	E,C,A	ESA



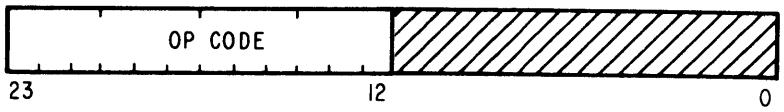
The state of the sign bit (A_{23}) of register A is copied into all 24 positions of register E and bit A_{23} is then set to zero. This forms a double-precision number in E and A.

0010.	Extend Sign of Byte	A,C	ESB



The state of the register B sign bit (A_7) is copied into bit positions A_8-A_{23}, forming a sign extension of the byte.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0054.	Floating NOrmalize	E,A,I,C	FNO



The contents of register D (E and A) are shifted left arithmetically until bit E_{22} differs from E_{23}. The negative shift count (i.e., the number of shifts performed) replaces the contents of register I.

Example: Convert a double-precision integer in D to double-precision floating point format.

```
TOC     0       Clear Overflow
FNO             Normalize
TIB             Position exponent in byte (A0-A7).
BOZ     *+2     If result is zero, no exponent adjustment
                is necessary.
AOB     46      Adjust shift count.
```

NOTES

There are four special cases where the shifting process differs from that described above.

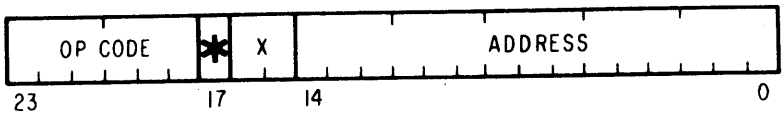
1. If the binary pattern 1000...0 is detected in register D, normalization is terminated to avoid creating the invalid pattern 10000...0.

2. If the invalid binary pattern 10000...0 is detected, it is shifted right one position, producting the pattern 11000...0. The shift count is adjusted accordingly.

3. If the pattern 00000...0 is detected, the shift count is set to -177_8, making a zero less significant than any other value.

4. If an Overflow condition is present when beginning the operation, the contents of register D are arithmetically shifted right one position. The shift count is set to ONE and the sign of D is complemented.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
56.*+X:a	MultiplY by Memory	E,A,C	MYM

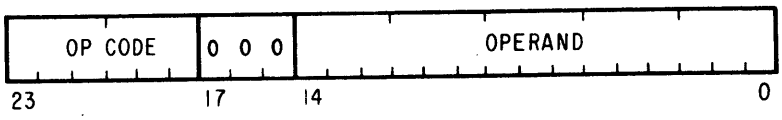


The contents of register A are algebraically multiplied by the contents of the effective memory address. The double-precision product replaces the previous contents of register D (E and A).

NOTE

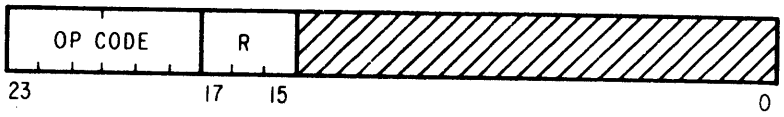
An Overflow will result if the full-scale negative number (1000....00) is used as both the multiplier and multiplicand.

60.0:o	MultiplY by Operand	E,A,C	MYO



The contents of register A are algebraically multiplied by the 15-bit unsigned operand in the instruction word. The double-precision product replaces the previous contents of register D (E and A).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
60.1	MultiplY by I	E,A,C	MYI
60.2	MultiplY by J	E,A,C	MYJ
60.3	MultiplY by K	E,A,C	MYK
60.4	MultiplY by E	E,A,C	MYE
60.5	MultiplY by A	E,A,C	MYA
60.6	MultiplY by T	E,A,C	MYT

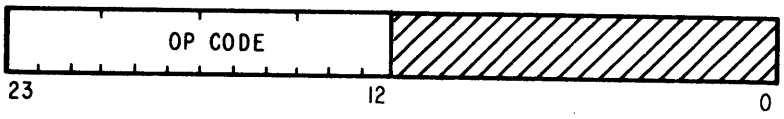


The contents of register A are algebraically multiplied by the contents of the specified register. The double-precision product replaces the previous contents of register D (E and A).

NOTE

An Overflow will result if the full-scale negative number (1000....00) is used as both the multiplier and multiplicand.

0005.	Negate of Byte to Byte	A,C	NBB

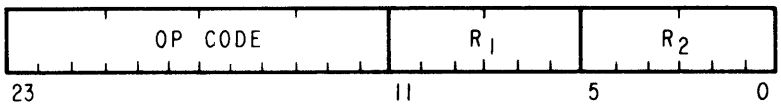


The contents of register B (A_0-A_7) are two's complemented. Bit positions A_8-A_{23} are unchanged.

NOTE

An Overflow will result when negating 2^7 (full-scale negative byte).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0022.01.01	Negate of I to I	I,C	NII
0022.01.02	Negate of I to J	J,C	NIJ
0022.01.04	Negate of I to K	K,C	NIK
0022.01.10	Negate of I to E	E,C	NIE
0022.01.20	Negate of I to A	A,C	NIA
0022.01.40	Negate of I to T	T,C	NIT
0022.02.01	Negate of J to I	I,C	NJI
0022.02.02	Negate of J to J	J,C	NJJ
0022.02.04	Negate of J to K	K,C	NJK
0022.02.10	Negate of J to E	E,C	NJE
0022.02.20	Negate of J to A	A,C	NJA
0022.02.40	Negate of J to T	T,C	NJT
0022.04.01	Negate of K to I	I,C	NKI
0022.04.02	Negate of K to J	J,C	NKJ
0022.04.04	Negate of K to K	K,C	NKK
0022.04.10	Negate of K to E	E,C	NKE
0022.04.20	Negate of K to A	A,C	NKA
0022.04.40	Negate of K to T	T,C	NKT
0022.10.01	Negate of E to I	I,C	NEI
0022.10.02	Negate of E to J	J,C	NEJ
0022.10.04	Negate of E to K	K,C	NEK
0022.10.10	Negate of E to E	E,C	NEE
0022.10.20	Negate of E to A	A,C	NEA
0022.10.40	Negate of E to T	T,C	NET
0022.20.01	Negate of A to I	I,C	NAI
0022.20.02	Negate of A to J	J,C	NAJ
0022.20.04	Negate of A to K	K,C	NAK
0022.20.10	Negate of A to E	E,C	NAE
0022.20.20	Negate of A to A	A,C	NAA
0022.20.40	Negate of A to T	T,C	NAT
0022.40.01	Negate of T to I	I,C	NTI
0022.40.02	Negate of T to J	J,C	NTJ
0022.40.04	Negate of T to K	K,C	NTK
0022.40.10	Negate of T to E	E,C	NTE
0022.40.20	Negate of T to A	A,C	NTA
0022.40.40	Negate of T to T	T,C	NTT

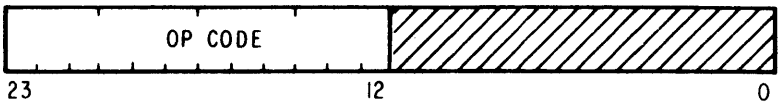


The two's complement of the contents of R_1 replace the previous contents of R_2.

NOTE

An Overflow will result when negating 2^{23} (full-scale negative number).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0033.	Negate of Double to Double	E,A,C	NDD



The contents of register D (E and A), in double-precision format, are two's complemented.

NOTE

An Overflow will result when negating 2^{46} (full-scale negative double integer).

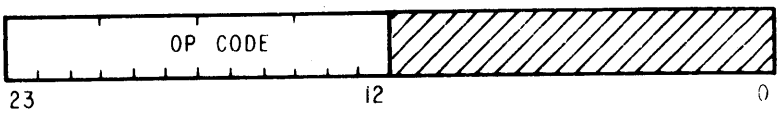
0032.01.01	Negate Sign of I	I,C	NSI
0032.02.02	Negate Sign of J	J,C	NSJ
0032.04.04	Negate Sign of K	K,C	NSK
0032.10.10	Negate Sign of E	E,C	NSE
0032.20.20	Negate Sign of A	A,C	NSA
0032.40.40	Negate Sign of T	T,C	NST

The sign bit of the specified register is complemented.

NOTE

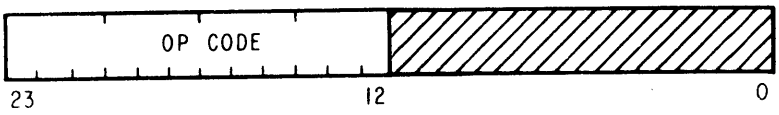
An Overflow will result when negating a full-scale negative.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0006.	Positive of Byte to Byte	A,C	PBB



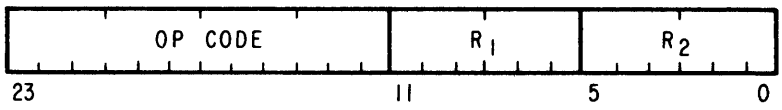
The absolute value of the contents of register B (A_0-A_7) is placed in register B.

0034.	Positive of Double to Double	E,A,C	PDD



The absolute value of the contents of register D is placed in register D according to the double integer format defined in Section II.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0023.01.01	Positive of I to I	I,C	PII
0023.01.02	Positive of I to J	J,C	PIJ
0023.01.04	Positive of I to K	K,C	PIK
0023.01.10	Positive of I to E	E,C	PIE
0023.01.20	Positive of I to A	A,C	PIA
0023.01.40	Positive of I to T	T,C	PIT
0023.02.01	Positive of J to I	I,C	PJI
0023.02.02	Positive of J to J	J,C	PJJ
0023.02.04	Positive of J to K	K,C	PJK
0023.02.10	Positive of J to E	E,C	PJE
0023.02.20	Positive of J to A	A,C	PJA
0023.02.40	Positive of J to T	T,C	PJT
0023.04.01	Positive of K to I	I,C	PKI
0023.04.02	Positive of K to J	J,C	PKJ
0023.04.04	Positive of K to K	K,C	PKK
0023.04.10	Positive of K to E	E,C	PKE
0023.04.20	Positive of K to A	A,C	PKA
0023.04.40	Positive of K to T	T,C	PKT
0023.10.01	Positive of E to I	I,C	PEI
0023.10.02	Positive of E to J	J,C	PEJ
0023.10.04	Positive of E to K	K,C	PEK
0023.10.10	Positive of E to E	E,C	PEE
0023.10.20	Positive of E to A	A,C	PEA
0023.10.40	Positive of E to T	T,C	PET
0023.20.01	Positive of A to I	I,C	PAI
0023.20.02	Positive of A to J	J,C	PAJ
0023.20.04	Positive of A to K	K,C	PAK
0023.20.10	Positive of A to E	E,C	PAE
0023.20.20	Positive of A to A	A,C	PAA
0023.20.40	Positive of A to T	T,C	PAT
0023.40.01	Positive of T to I	I,C	PTI
0023.40.02	Positive of T to J	J,C	PTJ
0023.40.04	Positive of T to K	K,C	PTK
0023.40.10	Positive of T to E	E,C	PTE
0023.40.20	Positive of T to A	A,C	PTA
0023.40.40	Positive of T to T	T,C	PTT

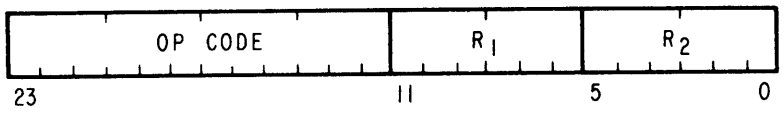


The absolute value of the contents of R_1 replaces the previous contents of R_2.

NOTE

An Overflow will result when negating a full-scale negative.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0075.01.01	Round of I to I	I,C	RII
0075.01.02	Round of I to J	J,C	RIJ
0075.01.04	Round of I to K	K,C	RIK
0075.01.10	Round of I to E	E,C	RIE
0075.01.20	Round of I to A	A,C	RIA
0075.01.40	Round of I to T	T,C	RIT
0075.02.01	Round of J to I	I,C	RJI
0075.02.02	Round of J to J	J,C	RJJ
0075.02.04	Round of J to K	K,C	RJK
0075.02.10	Round of J to E	E,C	RJE
0075.02.20	Round of J to A	A,C	RJA
0075.02.40	Round of J to T	T,C	RJT
0075.04.01	Round of K to I	I,C	RKI
0075.04.02	Round of K to J	J,C	RKJ
0075.04.10	Round of K to E	E,C	RKE
0075.04.20	Round of K to A	A,C	RKA
0075.04.40	Round of K to T	T,C	RKT
0075.10.01	Round of E to I	I,C	REI
0075.10.02	Round of E to J	J,C	REJ
0075.10.04	Round of E to K	K,C	REK
0075.10.10	Round of E to E	E,C	REE
0075.10.20	Round of E to A	A,C	REA
0075.10.40	Round of E to T	T,C	RET
0075.40.01	Round of T to I	I,C	RTI
0075.40.02	Round of T to J	J,C	RTJ
0075.40.04	Round of T to K	K,C	RTK
0075.40.10	Round of T to E	E,C	RTE
0075.40.20	Round of T to A	A,C	RTA
0075.40.40	Round of T to T	T,C	RTT

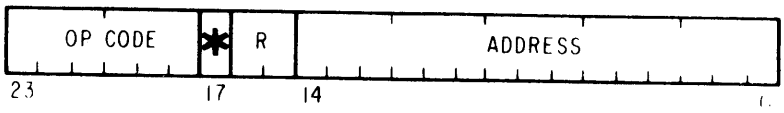


Round the contents of R_1 as a function of A and place the result in R_2.

NOTE

If bit A22 is a ONE, the contents of $R_1 + 1$ are transferred to R_2. If A22 is ZERO, the contents of R_1 replace the previous contents of R_2. In either case, R_1 is unchanged except when the same as R_2.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
51.*+2:a	Subtract Memory from I	I,C	SMI
51.*+2:a	Subtract Memory from J	J,C	SMJ
51.*+3:a	Subtract Memory from K	K,C	SMK



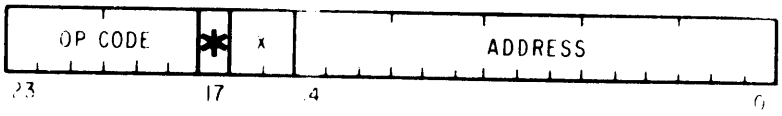
The contents of the effective memory address are algebraically subtracted from the contents of register I, J or K.

NOTE

The immediate memory reference cannot be indexed; however, indexing of indirect reference is permitted, e.g.,

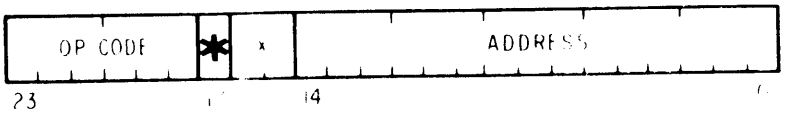
```
        SMI*    X
        .
        .
        .
X       DAC     Y,J
```

52.*+X:a	Subtract Memory from E	E,C	SME
53 *+X:a	Subtract Memory from A	A,C	SMA



The contents of the effective memory address are algebraically subtracted from the contents of register E or A.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
54.*+X:a	Subtract Memory from Double	E,A,C	SMD



The contents of the effective memory address (EMA) and the next sequential address (EMA+1) are algebraically subtracted from the contents of register D (E and A), according to the double integer format defined in Section II.

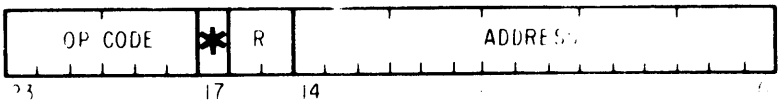
NOTE

Failure to adhere to the double integer format will provide unpredictable results. Bit A_{23} must be ZERO.

55.*+X:a	Subtract Memory from Byte	A,C	SMB

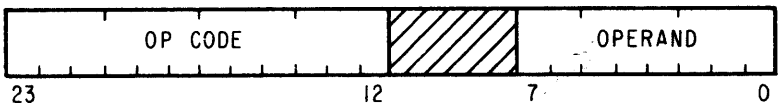
The contents of bits 0-7 of the effective memory address are algebraically subtracted from register B (A_0-A_7). Bits A_8-A_{23} are unaffected.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
65.1:o	Subtract Operand from I	I,C	SOI
65.2:o	Subtract Operand from J	J,C	SOJ
65.3:o	Subtract Operand from K	K,C	SOK
65.4:o	Subtract Operand from E	E,C	SOE
65.5:o	Subtract Operand from A	A,C	SOA
65.6:o	Subtract Operand from T	T,C	SOT



The 15-bit unsigned operand is algebraically subtracted from the contents of the specified register.

0013:o	Subtract Operand from Byte	A,C	SOB

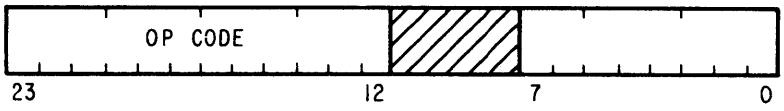


The 8-bit signed operand is algebraically subtracted from the contents of register B (A_0-A_7). Bits A_8-A_{23} are unaffected.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0021.01.02	Subtract I from J	J,C	SIJ
0021.01.04	Subtract I from K	K,C	SIK
0021.01.10	Subtract I from E	E,C	SIE
0021.01.20	Subtract I from A	A,C	SIA
0021.01.40	Subtract I from T	T,C	SIT
0021.02.01	Subtract J from I	I,C	SJI
0021.02.04	Subtract J from K	K,C	SJK
0021.02.10	Subtract J from E	E,C	SJE
0021.02.20	Subtract J from A	A,C	SJA
0021.02.40	Subtract J from T	T,C	SJT
0021.04.01	Subtract K from I	I,C	SKI
0021.04.02	Subtract K from J	J,C	SKJ
0021.04.10	Subtract K from E	E,C	SKE
0021.04.20	Subtract K from A	A,C	SKA
0021.04.40	Subtract K from T	T,C	SKT
0021.10.01	Subtract E from I	I,C	SEI
0021.10.02	Subtract E from J	J,C	SEJ
0021.10.04	Subtract E from K	K,C	SEK
0021.10.20	Subtract E from A	A,C	SEA
0021.10.40	Subtract E from T	T,C	SET
0021.20.01	Subtract A from I	I,C	SAI
0021.20.02	Subtract A from J	J,C	SAJ
0021.20.04	Subtract A from K	K,C	SAK
0021.20.10	Subtract A from E	E,C	SAE
0021.20.40	Subtract A from T	T,C	SAT
0021.40.01	Subtract T from I	I,C	STI
0021.40.02	Subtract T from J	J,C	STJ
0021.40.04	Subtract T from K	K,C	STK
0021.40.10	Subtract T from E	E,C	STE
0021.40.20	Subtract T from A	A,C	STA

The contents of R_1 are algebraically subtracted from R_2.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0076:014	Square Root	E,A,C	SRT
0076:027	Square Root — Extended	E,A,C	SRE



The contents of register A are treated as a 23-bit positive integer. The square root of this quantity is placed in register A, right justified, and the remainder is placed in register E so that:

$$root^2 + remainder = original\ integer.$$

If the sign bit (23) of register A is set, the Condition register will be set to OVERFLOW.

SRT generates a root of 12 significant bits; i.e., the true integer root of any positive integer in register A.

SRE generates a root of 23 significant bits. This extended significance is obtained by assuming 22 zeros to the right of bit A_0; effectively, multiplying the contents of A by 2^{22} and, consequently, the root by 2^{11}.

Consider the following examples where: A_n implies a binary point in the right of bit n.

Positve Integer	Instruction	Root (Octal)
2 at A_0	SRT	1 at A_0
2 at A_0	SRE	1.3240 at A_{11}
2 at A_{20}	SRT	1.3240 at A_{10}
2 at A_{20}	SRE	1.3240474 at A_{21}

7-6 BRANCH INSTRUCTIONS

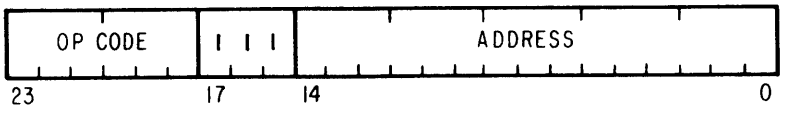
The Branch group of instructions can be divided into two basic types; conditional and unconditional branches. Conditional branches cause control to be transferred to a specified address upon detection of a certain machine condition as indicated by the contents of the Condition register. Unconditional branches cause control to be transferred unconditionally to a specified address.

In a non-virtual memory system only long branch (BJL, BLL, BRL, BSL, BUL) instructions or the Branch and Link —— Unrestricted (BLU) instruction should be used in the last location of the lower or upper 32K sections of memory. Use of any other Branch instruction will cause control to be automatically transferred to the opposite memory section.

The following instructions are included in the Branch group.

MNEMONIC	INSTRUCTION	PAGE
BBI	Branch when Byte Address +1 in I $\neq 0$	7-35
BBJ	Branch when Byte Address +1 in J $\neq 0$	7-35
BJL	Branch indexed by J—— Long	7-39
BLI	Branch and Link I	7-40
BLJ	Branch and Link J	7-40
BLK	Branch and Link K	7-40
BLL	Branch and Link (J) —— Long	7-40
BLU	Branch and Link —— Unrestricted	7-43
BNN	Branch on Not Negative	7-38
BNO	Branch on No Overflow	7-38
BNP	Branch on Not Positive	7-38
BNZ	Branch on Not Zero	7-38
BON	Branch on Negative	7-38
BOO	Branch on Overflow	7-38
BOP	Branch on Positive	7-38
BOZ	Branch on Zero	7-38
BRL	Branch on Reset interrupts —— Long	7-42
BSL	Branch and Save return —— Long	7-41
BUC	Branch UnConditionally	7-37
BUL	Branch Unconditionally —— Long	7-37
BWI	Branch When I+1 $\neq 0$	7-39
BWJ	Branch When J+1 $\neq 0$	7-39
BWK	Branch When K+1 $\neq 0$	7-39

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
60.7:a	Branch when Byte Address +1 in I $\neq 0$	I	BBI
61.7:a	Branch when Byte Address +1 in J $\neq 0$	J	BBJ



The contents of bits 22 and 23 of the specified index register (I or J) is incremented by one. If the result of this addition (in bits 22 and 23) is not 00_2, then the contents of register P (current PROGRAM ADDRESS) is replaced by the 15-bit effective memory address. If the result of the addition to bits 22 and 23 is 00_2, then bits 22 and 23 are set to 01_2 and bits 0-21 are incremented by one. If the resultant sum in bits 0-21 is zero, then register P advances to the next sequential program location and the index register is set to 20000000_8. Otherwise, the contents of register P are replaced by the 15-bit effective memory address.

In general, the BBI and BBJ instructions are used as special index register increments in order to sequentially reference consecutive bytes in memory via the EMB and RBM instructions. Consider the following example which will move 11 consecutive bytes starting from the third byte at location '200 to the first byte at location '300.

Example:

```
TMJ     = '60000200
TMI     = '20000300
TNK     11
EMB     0
RBM     0
BBI     *+1
BBJ     *+1
BWK     *-4
```

Occasionally, it is possible to use the address of portion of index register I or J as a byte counter as well as a word pointer. This may be illustrated by the following example which will set the buffer starting at byte 3 of location '100 through byte 3 of location '102 to blanks.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC

Example:

```
TOB     "Ø"
TMI     = '77777775 bits 22 and 23 = 3, bits 0-21 = -3
RBM     '100+3
BBI     *-1
```

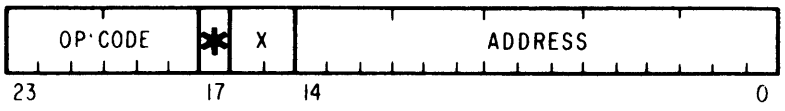
However, it should be noted this technique of using the index register as both a byte counter and word pointer may be used only in certain instances. Specifically, when the following relationship is true.

$$R\left(\frac{4-B.n.}{3}\right) = R\left(\frac{CT}{3}\right)$$

Where:

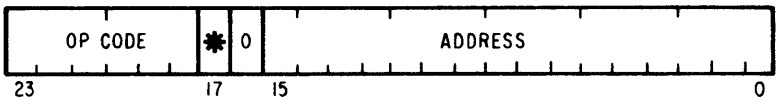
R ( )	= remainder
b.n.	= the starting byte number (1,2, or 3)
CT	= The number of bytes to be referenced.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
21.*+X:a	Branch UnConditionally	P	BUC



The contents of register P (current PROGRAM ADDRESS) are replaced by the 15-bit effective memory address.

26.*+0:A	Branch Unconditionally —— Long	P	BUL



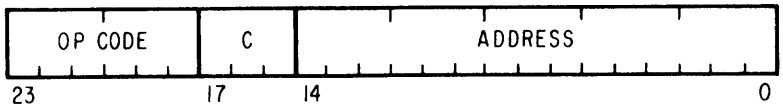
The contents of register P (current PROGRAM ADDRESS) are replaced by the 16-bit effective memory address.

NOTE

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,

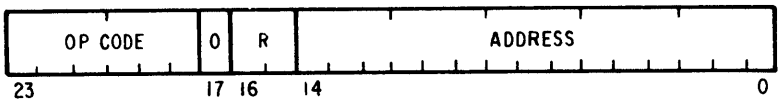
```
      BUC*   X
      .
      .
      .
X     DAC    Y,I
```

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
22.0:a	Branch On Overflow	P	BOO
22.1:a	Branch On Negative	P	BON
22.2:a	Branch On Zero	P	BOZ
22.3:a	Branch On Positive	P	BOP
22.4:a	Branch on No Overflow	P	BNO
22.5:a	Branch on Not Negative	P	BNN
22.6:a	Branch on Not Zero	P	BNZ
22.7:a	Branch on Not Positive	P	BNP



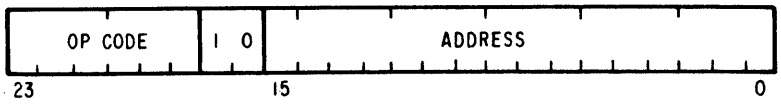
The contents of the Condition register are tested for the specified condition. If the condition is present, the contents of the P register (current PROGRAM ADDRESS) are replaced by the 16-bit effective memory address. If the specified condition is not present, the program advances to the next sequential location (PROGRAM ADDRESS +1).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
23.1:a	Branch When I+1 ≠ 0	I,P	BWI
23.2:a	Branch When J+1 ≠ 0	J,P	BWJ
23.3:a	Branch When K+1 ≠ 0	K,P	BWK



The contents of the specified register are incremented by one and then tested for zero. If the contents are not zero, the contents of register P (current PROGRAM ADDRESS) are replaced by the 16-bit effective memory address. If the contents are zero, the program advances to the next sequential location (PROGRAM ADDRESS +1).

23.4:A	Branch indexed by J —— Long	P	BJL

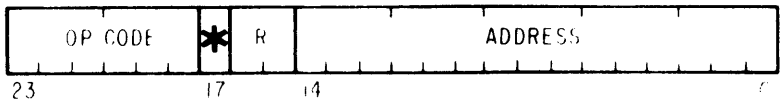


The contents of register P (current PROGRAM ADDRESS) are replaced by the 16-bit effective memory address.

NOTE

The immediate memory reference is automatically indexed by J.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
24.*+1:a	Branch and Link I	I,P	BLI
24.*+2:a	Branch and Link J	J,P	BLJ
24.*+3:a	Branch and Link K	K,P	BLK



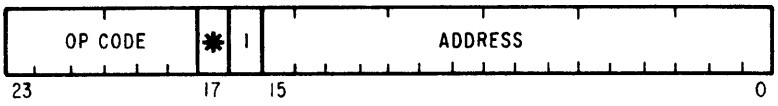
The contents of register I, J or K are replaced by the next sequential address (PROGRAM ADDRESS +1) and the contents of register P (current PROGRAM ADDRESS) are replaced by the 15-bit effective memory address.

NOTE

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.

```
        BLI*    X
        .
        .
        .
X       DAC     Y,J
```

26.*+2:A	Branch and Link (J) —— Long	J,P	BLL



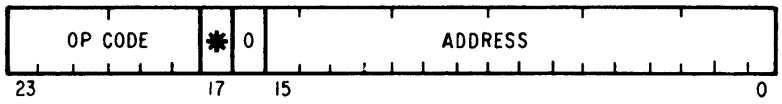
The contents of register J are replaced by the next sequential address (PROGRAM ADDRESS +1) and the contents of register P (current PROGRAM ADDRESS) are replaced by the 16-bit effective memory address.

NOTE

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.

```
        BLI*
        .
        .
        .
X       DAC
```

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
25.*+0:A	Branch and Save return —— Long	P	BSL



The next sequential address (PROGRAM ADDRESS +1), along with the contents of the Condition register are stored in the 16-bit effective memory address (EMA). The contents of register P (current PROGRAM ADDRESS) are then replaced by the address following the effective memory address (EMA +1).

This instruction is used to enter an interrupt subroutine because it provides a means of returning to the main program at the point of interrupt and saves the machine status (Condition) at the time of the interrupt.

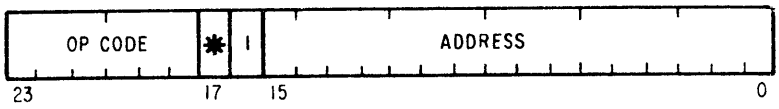
NOTES

1. The contents of the Condition register are stored in bit positions 16-19 of the EMA and the return address (PROGRAM ADDRESS +1) is stored in bits 0-15. The remaining bits are set to ZEROs; however, refer to the following for variation on bit 20.

2. The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted.

3. External interrupts are prohibited for the period of one instruction following the execution of this instruction.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
25.*+2:A	Branch and Reset interrupt —— Long	C,P	BRL



The highest-level active interrupt is reset (i.e., returned to the inactive state) and the contents of register P (current PROGRAM ADDRESS) are replaced by the 16-bit effective memory address.

BRL is normally used to exit an interrupt subroutine. If BRL contains an indirect reference, the last word in the indirect address chain contains the previous status (i.e., C register contents at the time of the interrupt) in bit positions M_{16}-M_{19} and the return address in bit positions M_0-M_{15} as a result of the BSL instruction. The C register is restored and the program branches to the return address (restarting the machine to the pre-interrupt status).

Example:

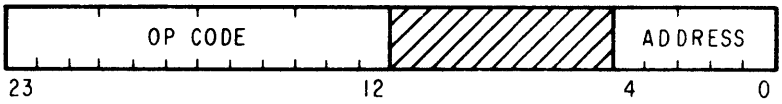
```
                .
                .
                TMA
L               AMA
                SMA             Interrupt occurs (EXM K).
                .
                .
K               BSL     M       Dedicated interrupt location.
M               ***             M becomes L+1 as a result of BSL
                :               at K. The C register contents
                :               are stored in M16-M19.
                BRL*    M       Restore C register and return to L+1.
```

NOTES

1. The BRL will not reset the interrupt if external interrupts have been held by an HXI instruction. Control will be returned to the effective memory address.
2. Those executive traps, which are not affected by the HXI instruction, will be reset by the BRL.
3. The immediate memory reference cannot be indexed; however, indexing indirect references is permitted, e.g.

```
                BRL*    X
                :
X               DAC     Y,K
```

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0067:a	Branch and Link —— Unrestricted	J,P	BLU



The next sequential address (PROGRAM ADDRESS +1) replaces the contents of register J and the contents of register P (current PROGRAM ADDRESS) are replaced by the 5-bit immediate memory address.

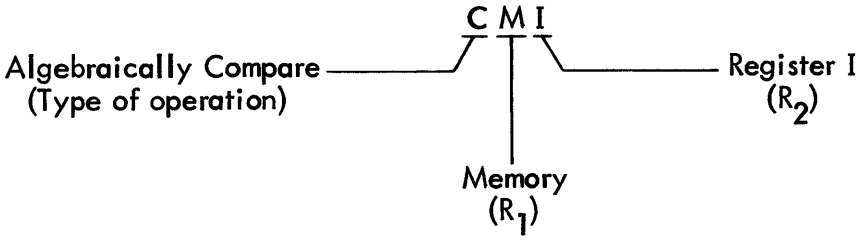
NOTES

1. Execution of the BLU instruction will turn OFF the Program Restricted Flag (PRF). If the computer is in a HALT condition and the PRF is ON, the BLU instruction will be treated as a NOP instruction.

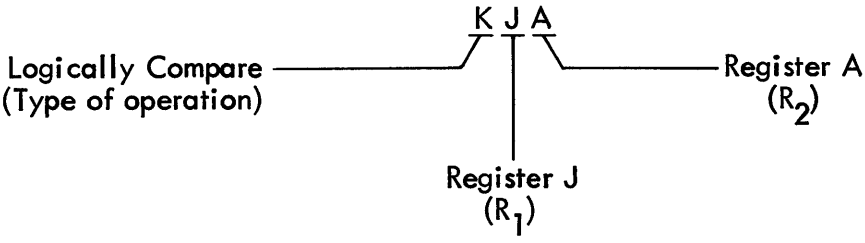
7-7 COMPARE INSTRUCTIONS

The Compare group of instructions is composed of two basic types of operations: algebraic and logical comparisons. Both types of instructions compare two referenced quantities and set the Condition register according to the result. Algebraic comparisons treat the references as signed (+ or -) quantities, while logical comparisons assume the references are unsigned quantities.

Algebraic comparisons are identified by the letter "C" as the first letter in the instruction mnemonic (e.g., CAI). Logical comparisons use a mnemonic code beginning with the letter "K" (KAI). The second letter of the mnemonic code designates the first of the compared quantities (R_1) and the last letter designates the second quantity (R_2). For example:



or



Both algebraic and logical comparisons are performed according to the formula:

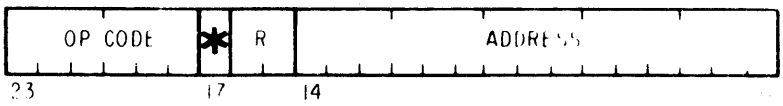
$$R_2 - R_1 = C \text{ (positive, zero or negative)}$$

Therefore, $R_2 > R_1$, $R_2 < R_1$ and $R_2 = R_1$ will set the Condition register (C) to positive (+), negative (-) and zero (0), respectively.

The following instructions are included in the SLASH 6 Compare group.

MNEMONIC	INSTRUCTION	PAGE
CAE	Compare A and E	7-51
CAI	Compare A and I	7-51
CAJ	Compare A and J	7-51
CAK	Compare A and K	7-51
CAT	Compare A and T	7-51

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
31.*+1:a	Compare Memory and I	C	CMI
31.*+2:a	Compare Memory and J	C	CMJ
31.*+3:a	Compare Memory and K	C	CMK



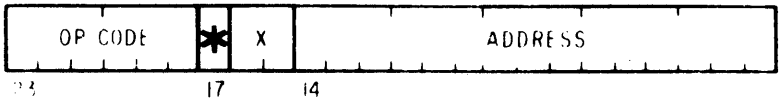
The contents of the effective memory address and the contents of register I, J, or K are algebraically compared and the Condition register is set to the status of the result.

NOTE

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.

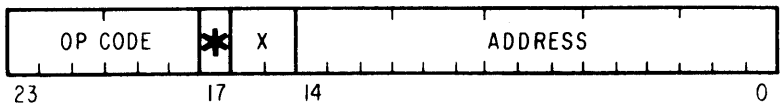
```
        CMI*    X
        .
        .
X       DAC     Y,K
```

32.*+X:a	Compare Memory and A	C	CMA
33.*+X:a	Compare Memory and E	C	CME



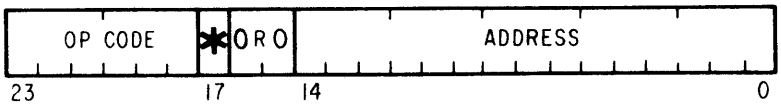
The contents of the effective memory address and the contents of register E or A are algebraically compared and the Condition register is set to the status of the result.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
34.*+X:a	Compare Memory and Byte	C	CMB



The contents of register B (A_0-A_7) and the contents of the effective memory address (M_0-M_7) are algebraically compared and the Condition register is set to the status of the result.

41.*+0:a	Compare Zero and Memory	C	CZM

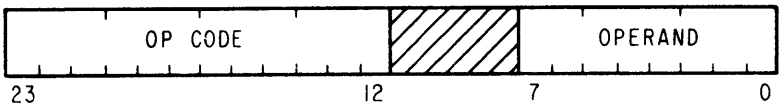


The contents of the effective memory address and zero are algebraically compared and the Condition register is set to the status of the result.

NOTE

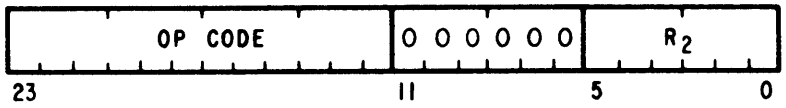
The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0014:o	Compare Operand and Byte	C	COB



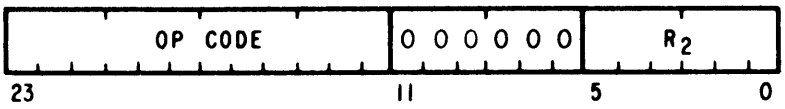
The 8-bit signed operand and the contents of register B (A_0-A_7) are algebraically compared and the Condition register is set to the status of the result.

0024.00.01	Compare Zero and I	C	CZI
0024.00.02	Compare Zero and J	C	CZJ
0024.00.04	Compare Zero and K	C	CZK
0024.00.10	Compare Zero and E	C	CZE
0024.00.20	Compare Zero and A	C	CZA
0024.00.40	Compare Zero and T	C	CZT



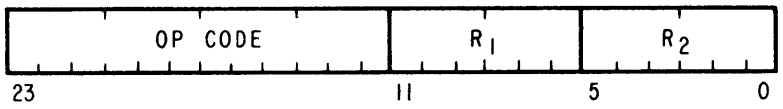
The contents of the specified register and zero are algebraically compared and the Condition register is set to the status of the result.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0024.00.30	Compare Zero and Double	C	CZD



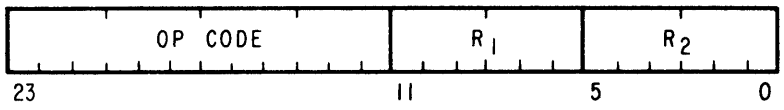
The contents of register D (E and A) and zero are algebraically compared and the Condition register is set to the status of the result.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0025.01.02	Compare I and J	C	CIJ
0025.01.04	Compare I and K	C	CIK
0025.01.10	Compare I and E	C	CIE
0025.01.20	Compare I and A	C	CIA
0025.01.40	Compare I and T	C	CIT
0024.02.01	Compare J and I	C	CJI
0024.02.04	Compare J and K	C	CJK
0024.02.10	Compare J and E	C	CJE
0024.02.20	Compare J and A	C	CJA
0024.02.40	Compare J and T	C	CJT
0024.04.01	Compare K and I	C	CKI
0024.04.02	Compare K and J	C	CKJ
0024.04.10	Compare K and E	C	CKE
0024.04.20	Compare K and A	C	CKA
0024.04.40	Compare K and T	C	CKT
0024.10.01	Compare E and I	C	CEI
0024.10.02	Compare E and J	C	CEJ
0024.10.04	Compare E and K	C	CEK
0024.10.20	Compare E and A	C	CEA
0024.10.40	Compare E and T	C	CET
0024.20.01	Compare A and I	C	CAI
0024.20.02	Compare A and J	C	CAJ
0024.20.04	Compare A and K	C	CAK
0024.20.10	Compare A and E	C	CAE
0024.20.40	Compare A and T	C	CAT
0024.40.01	Compare T and I	C	CTI
0024.40.02	Compare T and J	C	CTJ
0024.40.04	Compare T and K	C	CTK
0024.40.10	Compare T and E	C	CTE
0024.40.20	Compare T and A	C	CTA



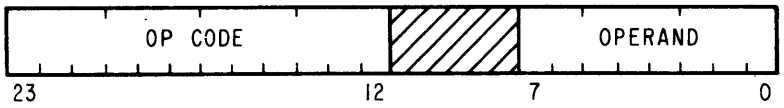
The contents of R_1 and the contents of R_2 are algebraically compared and the Condition register is set to the status of the result.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0025.01.02	Kompare I and J	C	KIJ
0025.01.04	Kompare I and K	C	KIK
0025.01.10	Kompare I and E	C	KIE
0025.01.20	Kompare I and A	C	KIA
0025.01.40	Kompare I and T	C	KIT
0025.02.01	Kompare J and I	C	KJI
0025.02.04	Kompare J and K	C	KJK
0025.02.10	Kompare J and E	C	KJE
0025.02.20	Kompare J and A	C	KJA
0025.02.40	Kompare J and T	C	KJT
0025.04.01	Kompare K and I	C	KKI
0025.04.02	Kompare K and J	C	KKJ
0025.04.10	Kompare K and E	C	KKE
0025.04.20	Kompare K and A	C	KKA
0025.04.40	Kompare K and T	C	KKT
0025.10.01	Kompare E and I	C	KEI
0025.10.02	Kompare E and J	C	KEJ
0025.10.04	Kompare E and K	C	KEK
0025.10.20	Kompare E and A	C	KEA
0025.10.40	Kompare E and T	C	KET
0025.20.01	Kompare A and I	C	KAI
0025.20.02	Kompare A and J	C	KAJ
0025.20.04	Kompare A and K	C	KAK
0025.20.10	Kompare A and E	C	KAE
0025.20.40	Kompare A and T	C	KAT
0025.40.01	Kompare T and I	C	KTI
0025.40.02	Kompare T and J	C	KTJ
0025.40.04	Kompare T and K	C	KTK
0025.40.10	Kompare T and E	C	KTE
0025.40.20	Kompare T and A	C	KTA



The contents of R_1 and R_2 are logically compared and the Condition register is set according to the result.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0015:o	Kompare Operand and Byte	C	KOB

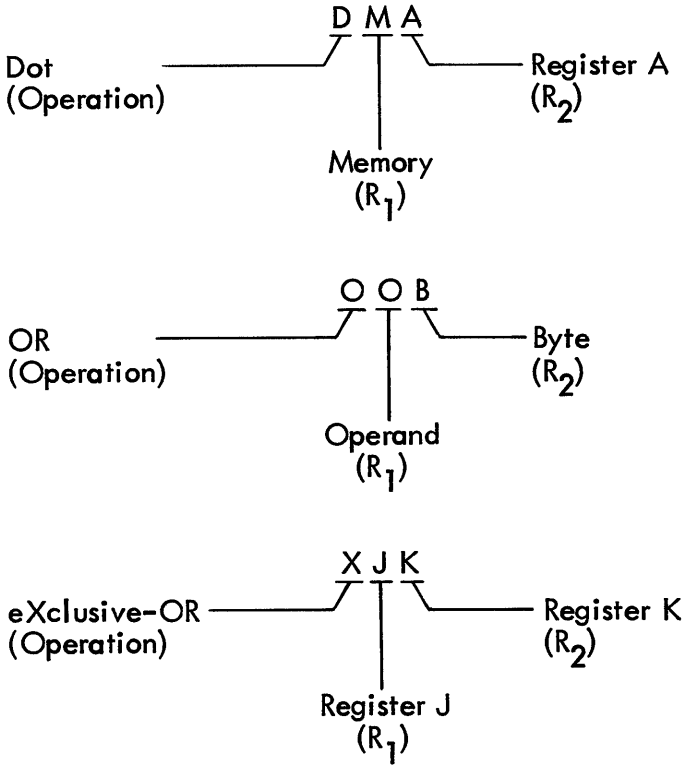


The 8-bit operand and the contents of register B (A_0-A_7) are logically compared and the Condition register is set according to the result.

7-8 LOGICAL INSTRUCTIONS

The Logical group of instructions includes AND (Dot product), OR and Exclusive-OR operations. All three types use two quantities to produce a logical result. The AND instructions use a mnemonic code beginning with the letter "D" for "Dot". The OR instructions use a mnemonic beginning with the letter "O", while Exclusive-OR instructions are distinguished by the letter "X".

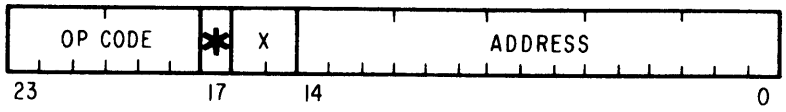
The second letter of the mnemonic code identifies the first of the two quantities (R_1). The third letter signifies the second quantity (R_2). Some examples are listed below.



Unless specifically noted otherwise in the individual descriptions, the result of the logical operation replaces the previous contents of R_2 while R_1 is unchanged. The Condition register is set to the status of the result (Positive, Negative, or Zero) after the operation. The various logical operations are illustrated in the following table.

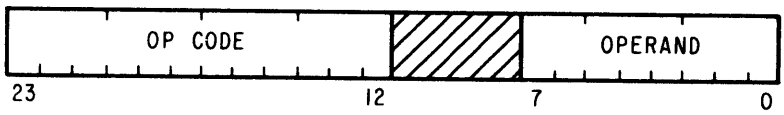
R_1	R_2	R_1 AND R_2	R_1 OR R_2	R_1 XOR R_2
1	1	1	1	0
0	1	0	1	1
1	0	0	1	1
0	0	0	0	0

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
36.*+X:a	Dot Memory with A	A,C	DMA



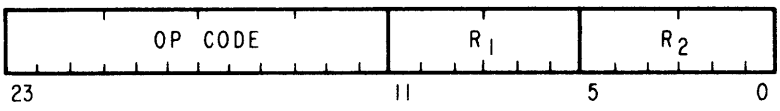
A logical AND is performed between the contents of the effective memory address and the contents of register A.

0016:o	Dot Operand with Byte	A,C	DOB



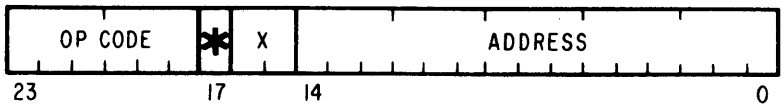
A logical AND is performed between the 8-bit operand and the contents of register B (A_0-A_7). Bits A_8-A_{23} are unchanged.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0026.01.02	Dot I with J	J,C	DIJ
0026.01.04	Dot I with K	K,C	DIK
0026.01.10	Dot I with E	E,C	DIE
0026.01.20	Dot I with A	A,C	DIA
0026.01.40	Dot I with T	T,C	DIT
0026.02.01	Dot J with I	I,C	DJI
0026.02.04	Dot J with K	K,C	DJK
0026.02.10	Dot J with E	E,C	DJE
0026.02.20	Dot J with A	A,C	DJA
0026.02.40	Dot J with T	T,C	DJT
0026.04.01	Dot K with I	I,C	DKI
0026.04.02	Dot K with J	J,C	DKJ
0026.04.10	Dot K with E	E,C	DKE
0026.04.20	Dot K with A	A,C	DKA
0026.04.40	Dot K with T	T,C	DKT
0026.10.01	Dot E with I	I,C	DEI
0026.10.02	Dot E with J	J,C	DEJ
0026.10.04	Dot E with K	K,C	DEK
0026.10.20	Dot E with A	A,C	DEA
0026.10.40	Dot E with T	T,C	DET
0026.20.01	Dot A with I	I,C	DAI
0026.20.02	Dot A with J	J,C	DAJ
0026.20.04	Dot A with K	K,C	DAK
0026.20.10	Dot A with E	E,C	DAE
0026.20.40	Dot A with T	T,C	DAT
0026.40.01	Dot T with I	I,C	DTI
0026.40.02	Dot T with J	J,C	DTJ
0026.40.04	Dot T with K	K,C	DTK
0026.40.10	Dot T with E	E,C	DTE
0026.40.20	Dot T with A	A,C	DTA



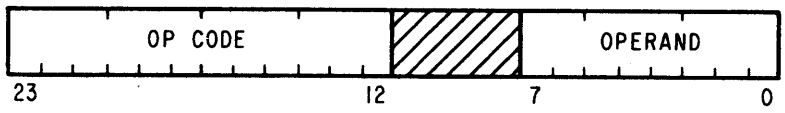
A logical AND is performed between the contents of R_1 and R_2.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
35.*+ X:a	Or Memory with A	A,C	OMA



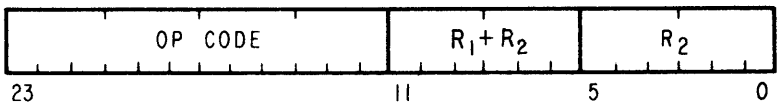
A logical OR is performed between the contents of the effective memory address and the contents of register A.

0004:o	Or Operand with Byte	A,C	OOB



A logical OR is performed between the 8-bit operand and the contents of register B (A_0-A_7). Bits A_8-A_{23} are unchanged.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0030.03.02	Or I with J	J,C	OIJ
0030.05.04	Or I with K	K,C	OIK
0030.11.10	Or I with E	E,C	OIE
0030.21.20	Or I with A	A,C	OIA
0030.41.40	Or I with T	T,C	OIT
0030.03.01	Or J with I	I,C	OJI
0030.06.04	Or J with K	K,C	OJK
0030.12.10	Or J with E	E,C	OJE
0030.22.20	Or J with A	A,C	OJA
0030.42.40	Or J with T	T,C	OJT
0030.05.01	Or K with I	I,C	OKI
0030.06.02	Or K with J	J,C	OKJ
0030.14.10	Or K with E	E,C	OKE
0030.24.20	Or K with A	A,C	OKA
0030.44.40	Or K with T	T,C	OKT
0030.11.01	Or E with I	I,C	OEI
0030.12.02	Or E with J	J,C	OEJ
0030.14.04	Or E with K	K,C	OEK
0030.30.20	Or E with A	A,C	OEA
0030.50.40	Or E with T	T,C	OET
0030.21.01	Or A with I	I,C	OAI
0030.22.02	Or A with J	J,C	OAJ
0030.24.04	Or A with K	K,C	OAK
0030.30.10	Or A with E	E,C	OAE
0030.60.40	Or A with T	T,C	OAT
0030.41.01	Or T with I	I,C	OTI
0030.42.02	Or T with J	J,C	OTJ
0030.44.04	Or T with K	K,C	OTK
0030.50.10	Or T with E	E,C	OTE
0030.60.20	Or T with A	A,C	OTA



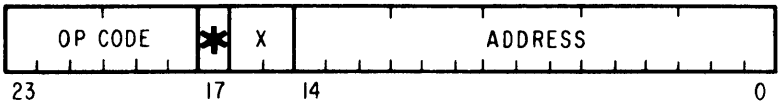
A logical OR is performed between the contents of R_1 and R_2.

NOTE

The general instruction formula for this group of Logical instructions is:

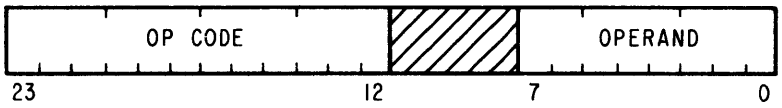
$$0030.R_1 + R_2 \cdot R_2$$

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
37.*+ X:a	eXclusive-or Memory with A	A,C	XMA



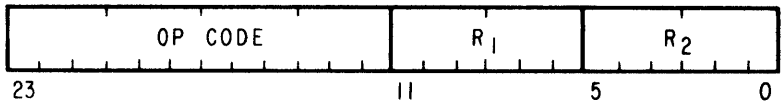
An EXCLUSIVE-OR operation is performed between the contents of the effective memory address and the contents of register A.

0017:o	eXclusive-or Operand with Byte	A,C	XOB



An EXCLUSIVE-OR operation is performed between the 8-bit operand and the contents of register B (A_0-A_7). Bits A_8-A_{23} are unchanged.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0027.01.02	eXclusive-or I with J	J,C	XIJ
0027.01.04	eXclusive-or I with K	K,C	XIK
0027.01.10	eXclusive-or I with E	E,C	XIE
0027.01.20	eXclusive-or I with A	A,C	XIA
0027.01.40	eXclusive-or I with T	T,C	XIT
0027.02.01	eXclusive-or J with I	I,C	XJI
0027.02.04	eXclusive-or J with K	K,C	XJK
0027.02.10	eXclusive-or J with E	E,C	XJE
0027.02.20	eXclusive-or J with A	A,C	XJA
0027.02.40	eXclusive-or J with T	T,C	XJT
0027.04.01	eXclusive-or K with I	I,C	XKI
0027.04.02	eXclusive-or K with J	J,C	XKJ
0027.04.10	eXclusive-or K with E	E,C	XKE
0027.04.20	eXclusive-or K with A	A,C	XKA
0027.04.40	eXclusive-or K with T	T,C	XKT
0027.10.01	eXclusive-or E with I	I,C	XEI
0027.10.02	eXclusive-or E with J	J,C	XEJ
0027.10.04	eXclusive-or E with K	K,C	XEK
0027.10.20	eXclusive-or E with A	A,C	XEA
0027.10.40	eXclusive-or E with T	T,C	XET
0027.20.01	eXclusive-or A with I	I,C	XAI
0027.20.02	eXclusive-or A with J	J,C	XAJ
0027.20.04	eXclusive-or A with K	K,C	XAK
0027.20.10	eXclusive-or A with E	E,C	XAE
0027.20.40	eXclusive-or A with T	T,C	XAT
0027.40.01	eXclusive-or T with I	I,C	XTI
0027.40.02	eXclusive-or T with J	J,C	XTJ
0027.40.04	eXclusive-or T with K	K,C	XTK
0027.40.10	eXclusive-or T with E	E,C	XTE
0027.40.20	eXclusive-or T with A	A,C	XTA



An Exclusive-or function is performed between the contents of R_1 and R_2.

7-9 SHIFT INSTRUCTIONS

The Shift instruction group consists of arithmetic and logical shifts. The arithmetic shifts cause the contents of a register to be shifted left or right a specified number of times, while preserving the original sign. The logical shifts are similar to the arithmetic shifts, except that the sign bit is shifted along with the other bits.

With both types of shift instructions, any number of shifts from 1 to 63 may be programmed without restriction. The number of shifts (n) are specified in bits 0-5 of the instruction word.

At the conclusion of any shitt operation, the Condition register is set to the status of the affected register's contents (Positive, Negative, Zero).

The time required to complete a Shift instruction is determined by the formula:

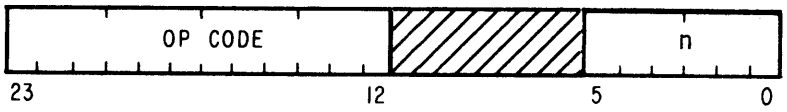
where $n > 5$	where $n \leq 5$
$t = 2+\left(\frac{n-1}{4}\right)$	$t = 2$

t = time in cycles
n = number of shifts

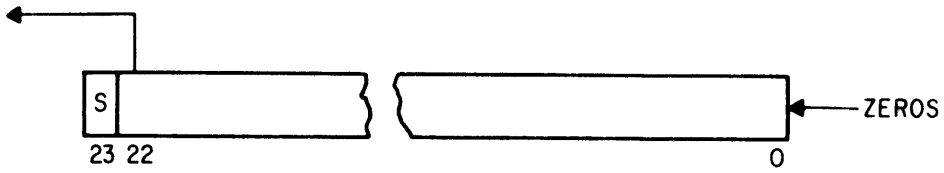
The following instructions are included in the Shift group.

MNEMONIC	INSTRUCTION	PAGE
LAA	Left shift Arithmetic A	7-64
LAD	Left shift Arithmetic Double	7-65
LLA	Left shift Logical A	7-66
LLD	Left shift Logical Double	7-66
LRA	Left Rotate A	7-67
LRD	Left Rotate Double	7-67
RAA	Right shift Arithmetic A	7-68
RAD	Right shift Arithmetic Double	7-68
RLA	Right shift Logical A	7-69
RLD	Right shift Logical Double	7-69
RRA	Right Rotate A	7-70
RRD	Right Rotate Double	7-70

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0040:n	Left shift Arithmetic A	A,C	LAA



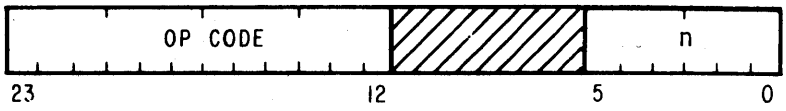
Bits A_{22}-A_0 are shifted left n places, with the most significant n bits being lost and n ZEROs being shifted into the least significant bit positions. The sign bit (A_{23}) is unchanged.



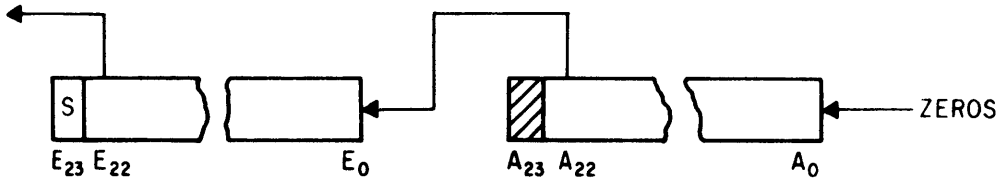
NOTE

If a bit shifted off from A_{22} differs from the sign bit, the Condition register will be set to OVERFLOW. (This is in addition to the Positive/Negative/Zero status.)

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0046:n	Left shift Arithmetic Double	E,A,C	LAD



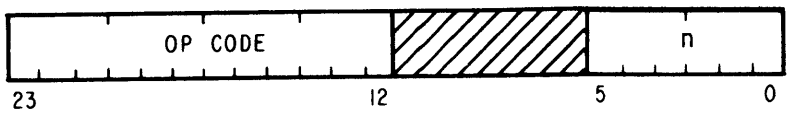
Bits E_{22}-E_0 and A_{22}-A_0 are shifted ——as one register —— left n places. The most significant n bits are lost and the least significant n bits are replaced with ZEROs. Bits E_{23} and A_{23} are bypassed. E_{23} is the register D sign bit and A_{23} is not used in the double-precision format.



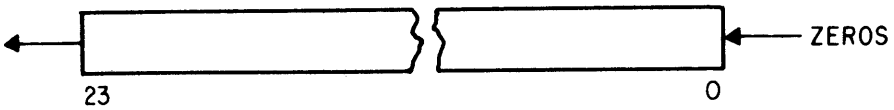
NOTE

If a bit shifted off from E_{22} differs from the sign bit, the Condition register will be set to OVERFLOW. (This is in addition to the Positive/Negative/Zero status.)

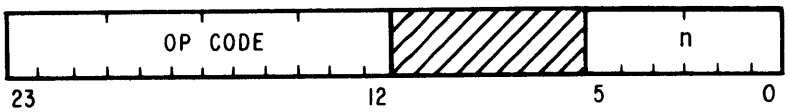
INSTRUCTION FORMULA		FUNCTION	REGISTERS AFFECTED	MNEMONIC
0042:n	L	Left shift Logical A	A,C	LLA



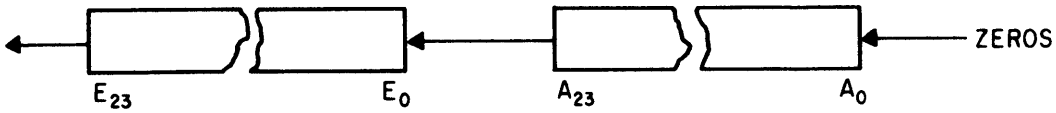
Bits A_{23}-A_0 are shifted left n places, with the most significant n bits being lost and the least significant n bits replaced by ZEROs.



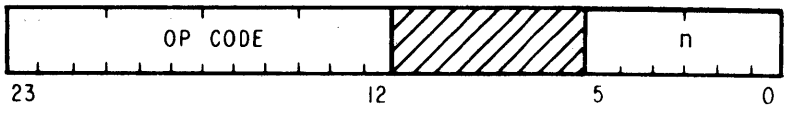
0050:n		Left shift Logical Double	E,A,C	LLD



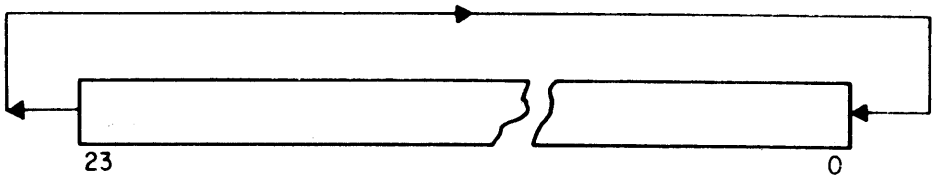
Bits E_{23}-E_0 and A_{23}-A_0 are shifted —— as one register —— left n places. The most significant n bits are lost and the least significant n bits are replaced with ZEROs.



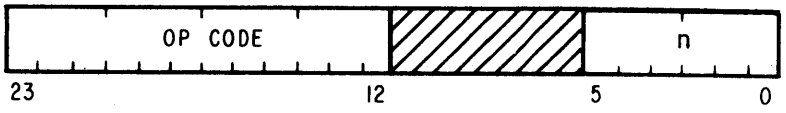
INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0044:n	Left Rotate A	A,C	LRA



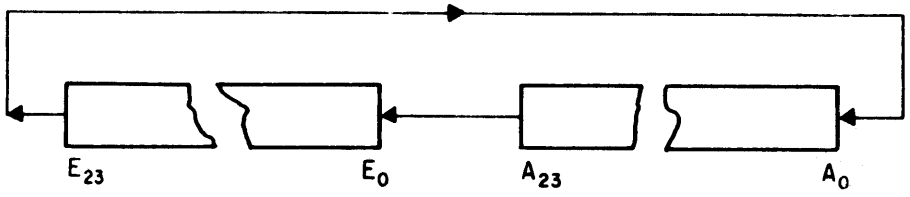
Bits A_{23}-A_0 are rotated left n places. No bits are lost.



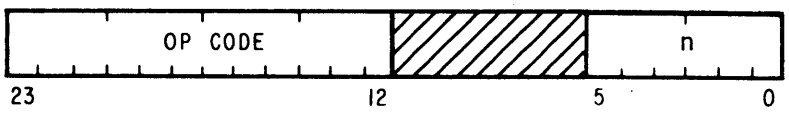
0052:n	Left Rotate Double	E,A,C	LRD



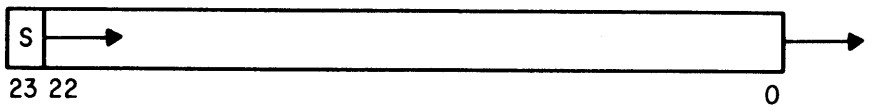
Bits E_{23}-E_0 and A_{23}-A_0 are rotated —— as one register —— left n places, with E_{23} replacing A_0 and A_{23} replacing E_0 as each shift takes place. No bits are lost.



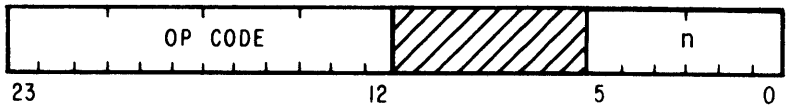
INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0041:n	Right shift Arithmetic A	A,C	RAA



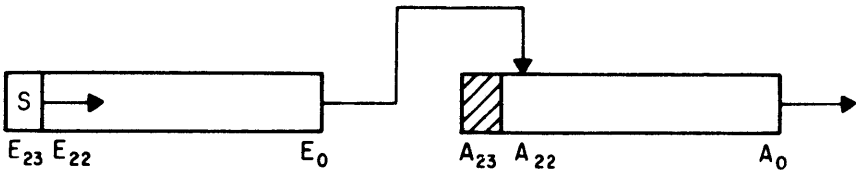
Bits A_{22}-A_0 are shifted right n places. The least significant n bits are lost and the most significant n bits are replaced by an extension of the sign bit (A_{23}). The sign bit is not changed.



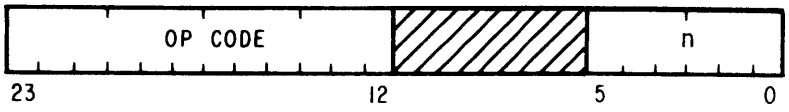
0047:n	Right shift Arithmetic Double	E,A,C	RAD



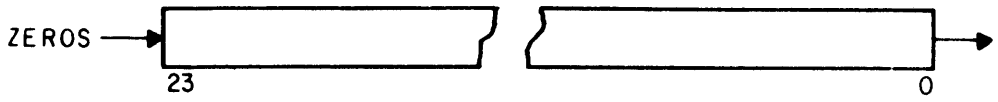
Bits E_{22}-E_0 and A_{22}-A_0 are shifted —— as one register —— right n places. The least significant n bits are lost and the most significant n bits are replaced by an extension of the sign bit (E_{23}). Bit A_{23} is bypassed.



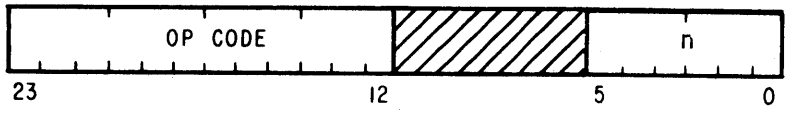
INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0043:n	Right shift Logical A	A,C	RLA



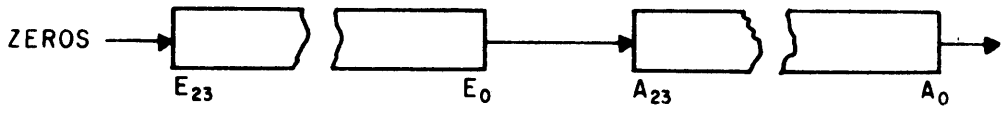
Bits A_{23}-A_0 are shifted right n places. The least significant n bits are lost and the most significant n bits are replaced by ZEROs.



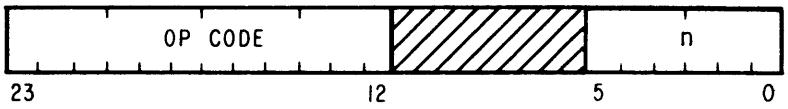
0051:n	Right shift Logical Double	E,A,C	RLD



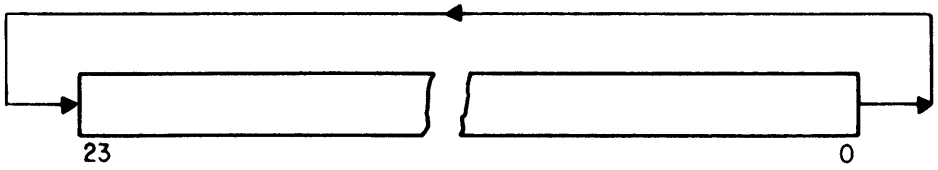
Bits E_{23}-E_0 and A_{23}-A_0 are shifted —— as one register —— right n places. The least significant n bits are lost and the most significant n bits are replaced by ZEROS.



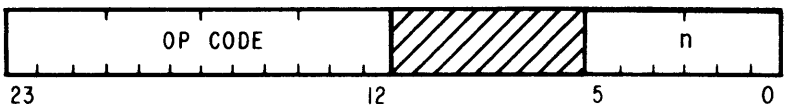
INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0045:n	Right Rotate A	A,C	RRA



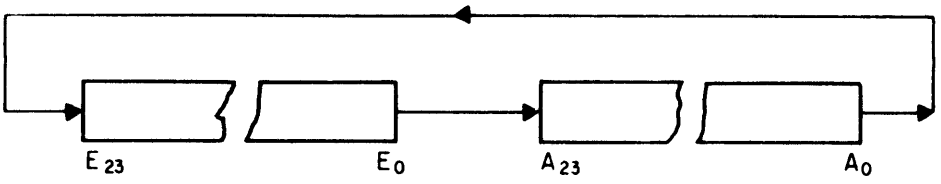
Bits A_{23}-A_0 are routed right n places. No bits are lost.



0053:n	Right Rotate Double	E,A,C	RRD



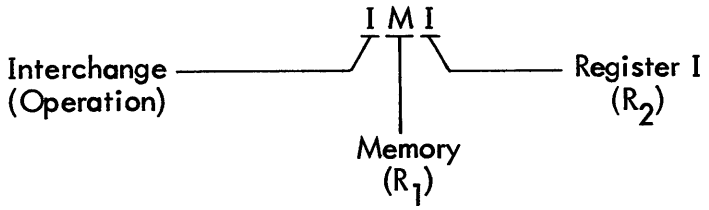
Bits E_{23}-E_0 and A_{23}-A_0 are rotated —— as one register —— right n places, with E_0 replacing A_{23} and A_0 replacing E_{23} as each shift takes place. No bits are lost.

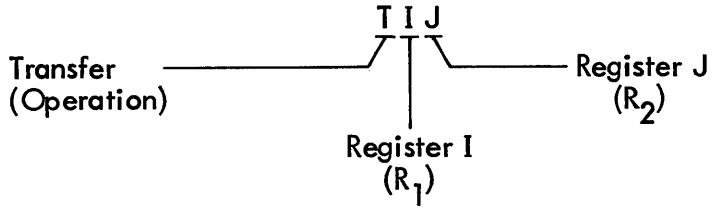


7-10 TRANSFER INSTRUCTIONS

The Transfer instruction group includes various types of operations. Among these are: interchanges between memory and a specified register, interchanges between registers, memory-to-register and register-to-memory transfers, and register-to-register transfers.

The mnemonic code for the Transfer instruction describes the individual operation. The first letter of the mnemonic indicates what action is to be taken; "I" for Interchange or "T" for Transfer. The second and third letters specify the source (R_1) and destination (R_2) respectively. Some examples are listed below:





With the exception of the interchange instructions, the transfer group (R_1) is not altered by the execution of any instructions in the Transfer group.

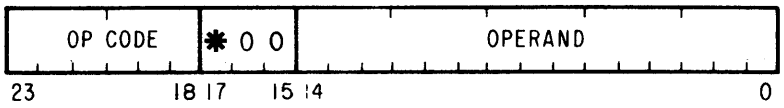
The Condition register is always set to reflect the status (Positive, Negative, or Zero) of the contents of R_2 at the completion of the instruction.

The following instructions are included in the Transfer group.

MNEMONIC	INSTRUCTION	PAGE
EMB	Extract Memory Byte	7-75
IAE	Interchange A and E	7-77
IAI	Interchange A and I	7-77
IAJ	Interchange A and J	7-77
IAK	Interchange A and K	7-77
IAT	Interchange A and T	7-77
IEA	Interchange E and A	7-77
IEI	Interchange E and I	7-77
IEJ	Interchange E and J	7-77
IEK	Interchange E and K	7-77
IET	Interchange E and T	7-77

MNEMONIC	INSTRUCTION	PAGE
IIA	Interchange I and A	7-77
IIE	Interchange I and E	7-77
IIJ	Interchange I and J	7-77
IIK	Interchange I and K	7-77
IIT	Interchange I and T	7-77
IJA	Interchange J and A	7-77
IJE	Interchange J and E	7-77
IJI	Interchange J and I	7-77
IJK	Interchange J and K	7-77
IJT	Interchange J and T	7-77
IKA	Interchange K and A	7-77
IKE	Interchange K and E	7-77
IKI	Interchange K and I	7-77
IKJ	Interchange K and J	7-77
IKT	Interchange K and T	7-77
IMA	Interchange M and A	7-76
IME	Interchange M and E	7-76
IMI	Interchange M and I	7-76
IMJ	Interchange M and J	7-76
IMK	Interchange M and K	7-76
ITA	Interchange T and A	7-77
ITE	Interchange T and E	7-77
ITI	Interchange T and I	7-77
ITJ	Interchange T and J	7-77
ITK	Interchange T and K	7-77
RBM	Replace Byte in Memory	7-78
TAE	Transfer A to E	7-89
TAI	Transfer A to I	7-89
TAJ	Transfer A to J	7-89
TAK	Transfer A to K	7-89
TAM	Transfer A to Memory	7-88
TAT	Transfer A to T	7-89
TBM	Transfer Byte to Memory	7-86
TDM	Transfer Double to Memory	7-86
TEA	Transfer E to A	7-89
TEB	Transfer E to B	7-85
TEI	Transfer E to I	7-89
TEJ	Transfer E to J	7-89
TEK	Transfer E to K	7-89
TEM	Transfer E to Memory	7-88
TET	Transfer E to T	7-89
TFM	Transfer Flag to Memory	7-87
TIA	Transfer I to A	7-89
TIB	Transfer I to Byte	7-85
TIE	Transfer I to E	7-89
TIJ	Transfer I to J	7-89
TIK	Transfer I to K	7-89
TIM	Transfer I to Memory	7-88
TIT	Transfer I to T	7-89
TJA	Transfer J to A	7-89
TJB	Transfer J to Byte	7-85

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
31.*+0:a	Extract Memory Byte	B,C	EMB



The effective memory address is added to the contents of register J, producing the word address which contains the byte to be extracted. The selected byte, as determined by the contents of bits 22 and 23 of the index register J, is then placed in register B. The following table shows the correspondence between bits 22 and 23 of J and the byte to be extracted:

Bits 22 and 23 J Register	Byte Selection
01	Leftmost byte (bits 16-23 of EMA+J)
10	Middle byte (bits 8-15 of EMA+J)
11	Rightmost byte (bits 0-7 of EMA+J)
00	Rightmost byte (bits 0-7 of EMA+J)

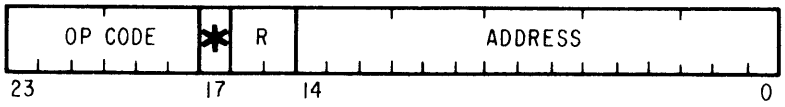
The final address of any indirect index sequence should not be indexed since implied indexing on register J takes place. If indexing is specified on the final address, then the specified index register will be logically ORed with register J prior to the add function with the EMA.

Examples:

```
            If J = '40000030
            and K = '00000010 when the following is executed:
            EMB*    '40
'40         DAC*    '50,K
'42         DATA    "XYZ"
'60         DAC     '12
```

Then the character Y will be placed in register B. Note that the effective address of the indirect/index sequence is '12. However, '12 plus bits 0-15 of index register J ('30) yields the final address of '42. Since a byte specification of 10_2 was made in bits 22-23 of index register J, then the second byte (bits 8-15) of memory location '42 is placed in register B.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
66.*+1:a	Interchange Memory and I	M,I,C	IMI
66.*+2:a	Interchange Memory and J	M,J,C	IMJ
66.*+3:a	Interchange Memory and K	M,K,C	IMK



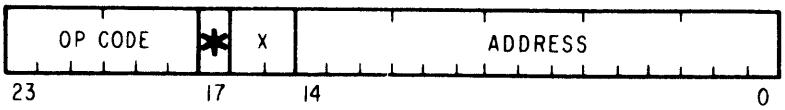
The contents of the effective memory address and register I, J, or K are interchanged.

NOTE

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.

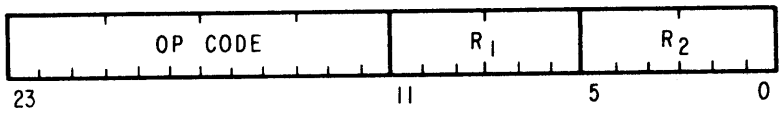
```
        IMK*    X
        .
        .
X       DAC     Y,J
```

67.*+X:a	Interchange Memory and E	M,E,C	IME
70.*+X:a	Interchange Memory and A	M,A,C	IMA



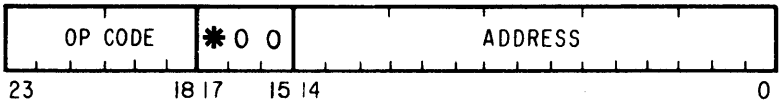
The contents of the effective memory address and the specified register are interchanged.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0035.01.02	Interchange I and J	I,J,C	IIJ
0035.01.04	Interchange I and K	I,K,C	IIK
0035.01.10	Interchange I and E	I,E,C	IIE
0035.01.20	Interchange I and A	I,A,C	IIA
0035.01.40	Interchange I and T	I,T,C	IIT
0035.02.01	Interchange J and I	J,I,C	IJI
0035.02.04	Interchange J and K	J,K,C	IJK
0035.02.10	Interchange J and E	J,E,C	IJE
0035.02.20	Interchange J and A	J,A,C	IJA
0035.02.40	Interchange J and T	J,T,C	IJT
0035.04.01	Interchange K and I	K,T,C	IKI
0035.04.02	Interchange K and J	K,J,C	IKJ
0035.04.10	Interchange K and E	K,E,C	IKE
0035.04.20	Interchange K and A	K,A,C	IKA
0035.04.40	Interchange K and T	K,T,C	IKT
0035.10.01	Interchange E and I	E,I,C	IEI
0035.10.02	Interchange E and J	E,J,C	IEJ
0035.10.04	Interchange E and K	E,K,C	IEK
0035.10.20	Interchange E and A	E,A,C	IEA
0035.10.40	Interchange E and T	E,T,C	IET
0035.20.01	Interchange A and I	A,I,C	IAI
0035.20.02	Interchange A and J	A,J,C	IAJ
0035.20.04	Interchange A and K	A,K,C	IAK
0035.20.10	Interchange A and E	A,E,C	IAE
0035.20.40	Interchange A and T	A,T,C	IAT
0035.40.01	Interchange T and I	T,I,C	ITI
0035.40.02	Interchange T and J	T,J,C	ITJ
0035.40.04	Interchange T and K	T,K,C	ITK
0035.40.10	Interchange T and E	T,E,C	ITE
0035.40.20	Interchange T and A	T,A,C	ITA



The contents of R_1 and R_2 are interchanged.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
27.*+0:a	Replace Byte in Memory	M	RBM

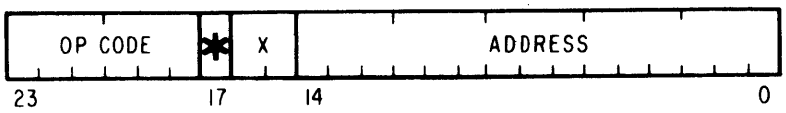


The effective memory address is added to the contents of register I producing the word address which contains the byte to be replaced. The selected byte, as determined by the contents of bits 22 and 23 of index register I, is then replaced by the contents of register B. The following table shows the correspondence between bits 22 and 23 of I and the byte to be replaced.

Bits 23 and 22 I Register	Byte Selection
01	Leftmost byte (bits 16-23 of EMA+I)
10	Middle byte (bits 8-15 of EMA+I)
11	Rightmost byte (bits 0-7 of EMA+I)
00	Causes no operation

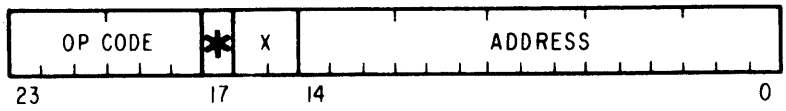
The final address of any indirect/index sequence should not be indexed since implied indexing on register I takes place. If indexing is specified on the final address, then the specified index register will be logically ORed with register I prior to the add function with the EMA.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
07.*+X:a	Transfer Memory to Byte	A,C	TMB



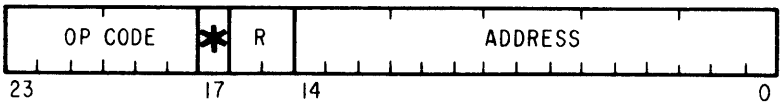
The 8 least significant bits (0-7) of the contents of the effective memory address replace the previous contents of register B (A_0-A_7). Bits A_8-A_{23} are unaffected.

06.*+X:a	Transfer Memory to Double	E,A,C	TMD



The contents of the effective memory address (EMA) and the next sequential address (EMA+1) replace the previous contents of register D (E and A). EMA and EMA+1 are transferred to E and A, respectively.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
51.*+0:a	Transfer Memory to Query register	Q	TMQ



Bits 0-17, 21, 22 and 23 of the contents of the effective memory address replace the previous contents of the Query register. These bits are loaded into the Q register in bit positions 0-17, 18, 19, and 20, respectively.

Executing this instruction will cause the Program Halt and Address Trap to be enabled or disabled, depending on the state of bits 23, 22, and 21 of the effective memory address.

Bit 23 = ONE = Disable Address Trap
Bit 23 = ZERO = Enable Address Trap

Bit 22 = ONE = Trap on Write only
Bit 22 = ZERO = Trap each time selected address is referenced

Bit 21 = ONE = Trap or Halt during User Mode only
Bit 21 = ZERO = Trap or Halt during Monitor mode only

Example:

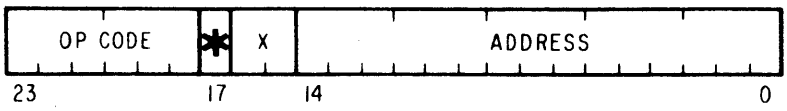
```
        TMQ         OA
        ...
OA      DAC         ADDR        Enable Address Trap
        or
OA      DAC*        O           Disable Address Trap
```

NOTE

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,

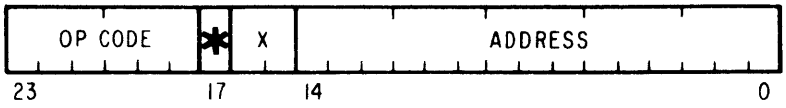
```
            TMQ*    X
            .
            .
X           DAC     Y,I
```

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
01.*+X:a	Transfer Memory to I	I,C	TMI
02.*+X:a	Transfer Memory to J	J,C	TMJ
03.*+X:a	Transfer Memory to K	K,C	TMK
04.*+X:a	Transfer Memory to E	E,C	TME
05.*+X:a	Transfer Memory to A	A,C	TMA



The contents of the effective memory address replace the previous contents of the specified register.

10.*+X:a	Transfer Memory to Registers	I,J,K,E,A	TMR

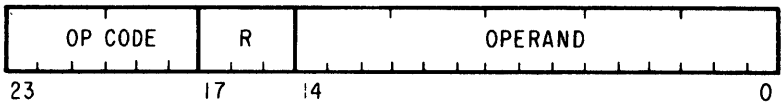


Registers I, J, K, E and A are loaded from consecutive memory addresses beginning with the effective memory address.

NOTE

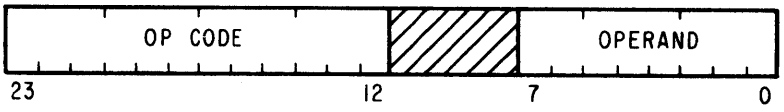
External interrupts are prohibited for the period of one instruction following the execution of this instruction.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
63.1:o	Transfer Negative operand to I	I,C	TNI
63.2:o	Transfer Negative operand to J	J,C	TNJ
63.3:o	Transfer Negative operand to K	K,C	TNK
63.4:o	Transfer Negative operand to E	E,C	TNE
63.5:o	Transfer Negative operand to A	A,C	TNA
63.6:o	Transfer Negative operand to T	T,C	TNT



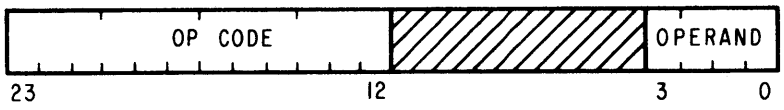
The two's complement of the 15-bit unsigned operand replaces the previous contents of bits 0-23 of the specified register.

0003:o	Transfer Operand to Byte	A,C	TOB



The 8-bit operand replaces the previous contents of register B (A_0-A_7). Bits A_8-A_{23} are unaffected.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0036:o	Transfer Operand to Condition register	C	TOC

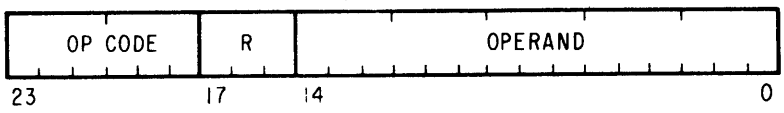


The 4-bit operand replaces the previous contents of the Condition register.

Operand definition is as follows:

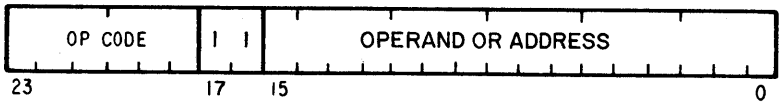
bit 0 - ONE = Overflow,	ZERO = No Overflow
bit 1 - ONE = Negative,	ZERO = Not Negative
bit 2 - ONE = Zero,	ZERO = Not Zero
bit 3 - ONE = Positive,	ZERO = Not Positive

62.1:o	Transfer Operand to I	I,C	TOI
62.2:o	Transfer Operand to J	J,C	TOJ
62.3:o	Transfer Operand to K	K,C	TOK
62.4:o	Transfer Operand to E	E,C	TOE
62.5:o	Transfer Operand to A	A,C	TOA
62.6:o	Transfer Operand to T	T,C	TOT



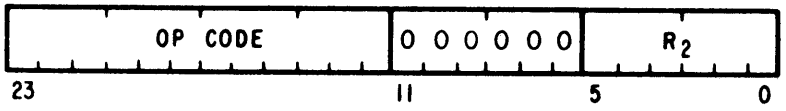
The 15-bit operand replaces the previous contents of bits 0-23 of the specified register.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
23.6:A	Transfer Long Operand to K	K	TLO



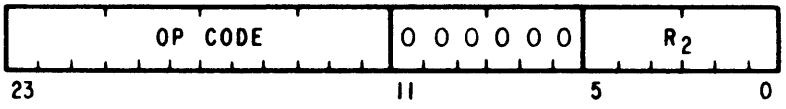
The 16-bit operand (or address) replaces the previous contents of bits 0-23 of register K.

0031.00.01	Transfer Switches to I	I,C	TSI
0031.00.02	Transfer Switches to J	J,C	TSJ
0031.00.04	Transfer Switches to K	K,C	TSK
0031.00.10	Transfer Switches to E	E,C	TSE
0031.00.20	Transfer Switches to A	A,C	TSA
0031.00.40	Transfer Switches to T	T,C	TST



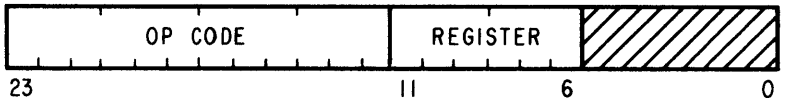
The states (set = ONE) of the console control switches (i.e., switch register) are transferred to the corresponding bit positions of the specified register.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0030.00.01	Transfer Zero to I	I,C	TZI
0030.00.02	Transfer Zero to J	J,C	TZJ
0030.00.04	Transfer Zero to K	K,C	TZK
0030.00.10	Transfer Zero to E	E,C	TZE
0030.00.20	Transfer Zero to A	A,C	TZA
0030.00.40	Transfer Zero to T	T,C	TZT
0030.00.30	Transfer Zero to Double	E,A,C	TZD



The previous contents of the specified register are replaced with ZEROs.

0002.01	Transfer I to Byte	A	TIB
0002.02	Transfer J to Byte	A	TJB
0002.04	Transfer K to Byte	A	TKB
0002.10	Transfer E to Byte	A	TEB
0002.40	Transfer T to Byte	A	TTB

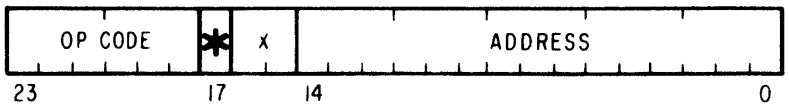


The least significant 8 bits (0-7) of the contents of the specified register replace the previous contents of register B (A_0-A_7). Bits A_8-A_{23} are unchanged.

NOTE

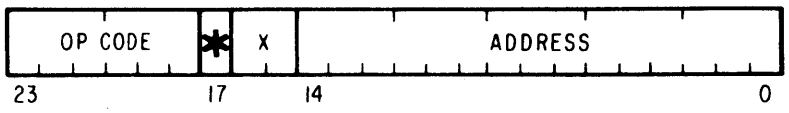
The Condition register is not affected.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
17.*+X:a	Transfer Byte to Memory	M	TBM



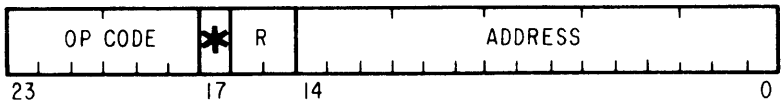
The contents of register B (A_0-A_7) replace the 8 least significant bits (0-7) of the contents of the effective memory address. Bits 8-23 of the memory word are unaffected.

16.*+X:a	Transfer Double to Memory	M	TDM



The contents of register D (E and A) replace the previous contents of the effective memory address (EMA) and the next sequential address (EMA+1). The contents of E and A are transferred to EMA and EMA+1, respectively.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
46.*+0:a	Transfer Flag to Memory	M,C	TFM



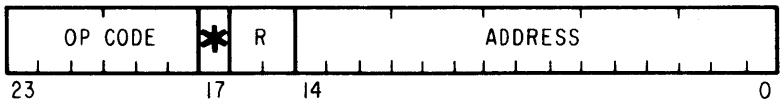
The previous contents of the effective memory address are replaced by ONEs.

NOTES

1. The Condition (C) register is set to the status of memory prior to the transfer.
2. The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,

```
        TFM*    X
        .
        .
X       DAC     Y,I
```

66.*+0:a Transfer Zero to Memory M,C TZM



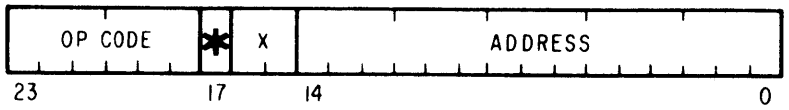
The previous contents of the effective memory address are replaced by ZEROs.

NOTES

1. The Condition (C) register is set to the status of memory prior to the transfer.
2. The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,

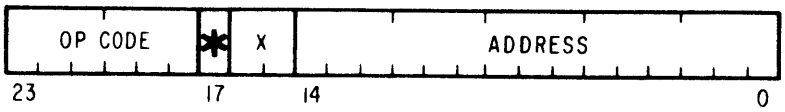
```
        TZM*    X
        .
        .
X       DAC     Y,I
```

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
11.*+X:a	Transfer I to Memory	M	TIM
12.*+X:a	Transfer J to Memory	M	TJM
13.*+X:a	Transfer K to Memory	M	TKM
14.*+X:a	Transfer E to Memory	M	TEM
15.*+X:a	Transfer A to Memory	M	TAM



The contents of the specified register replace the previous contents of the effective memory address.

20.*+X:a	Transfer Registers to Memory	M	TRM

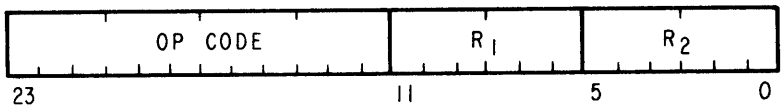


The contents of registers I, J, K, E and A are stored in consecutive memory locations beginning with the effective memory address.

NOTE

External interrupts are prohibited for the period of one instruction following the execution of this instruction.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0030.01.02	Transfer I to J	J,C	TIJ
0030.01.04	Transfer I to K	K,C	TIK
0030.01.10	Transfer I to E	E,C	TIE
0030.01.20	Transfer I to A	A,C	TIA
0030.01.40	Transfer I to T	T,C	TIT
0030.02.01	Transfer J to I	I,C	TJI
0030.02.04	Transfer J to K	K,C	TJK
0030.02.10	Transfer J to E	E,C	TJE
0030.02.20	Transfer J to A	A,C	TJA
0030.02.40	Transfer J to T	T,C	TJT
0030.04.01	Transfer K to I	I,C	TKI
0030.04.02	Transfer K to J	I,C	TKJ
0030.04.10	Transfer K to E	E,C	TKE
0030.04.20	Transfer K to A	A,C	TKA
0030.04.40	Transfer K to T	T,C	TKT
0030.10.01	Transfer E to I	I,C	TEI
0030.10.02	Transfer E to J	J,C	TEJ
0030.10.04	Transfer E to K	K,C	TEK
0030.10.20	Transfer E to A	A,C	TEA
0030.10.40	Transfer E to T	T,C	TET
0030.20.01	Transfer A to I	I,C	TAI
0030.20.02	Transfer A to J	J,C	TAJ
0030.20.04	Transfer A to K	K,C	TAK
0030.20.10	Transfer A to E	E,C	TAE
0030.20.40	Transfer A to T	T,C	TAT
0030.40.01	Transfer T to I	I,C	TTI
0030.40.02	Transfer T to J	J,C	TTJ
0030.40.04	Transfer T to K	K,C	TTK
0030.40.10	Transfer T to E	E,C	TTE
0030.40.20	Transfer T to A	A,C	TTA



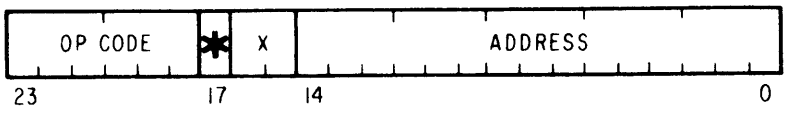
The contents of R_1 replace the previous contents of R_2.

7-11 BYTE PROCESSING INSTRUCTIONS

The Byte Processing group of instructions permits program manipulation of all three bytes within the computer word (24 bits); e. g., extract, replace, etc. The following instructions are inclusive of byte processing operations:

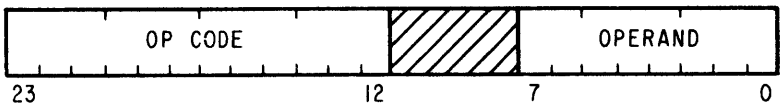
MNEMONIC	INSTRUCTION	PAGE
AMB	Add Memory to Byte	7-91
AOB	Add Operand to Byte	7-91
BBI	Brnach when Byte address +1 in I $\neq$ 0	8-92
BBJ	Branch when Byte address in +1 in J $\neq$ 0	7-92
CMB	Compa re Memory and Byte	7-93
COB	Compare Operand and Byte	7-94
DOB	Dot Operand with Byte	7-94
EMB	Extract Memory Byte	7-95
ESB	Extend Sign of Byte	7-96
EZB	Extend Zeros from Byte	7-96
KOB	Kompare Operand and Byte	7-97
NBB	Negate of Byte to Byte	7-97
OOB	Or Operand with Byte	7-98
PBB	Positive of Byte to Byte	7-98
RBM	Replace Byte in Memory	7-99
QBB	Query Bits of Byte	7-100
SOB	Subtract Operand from Byte	7-101
TBM	Transfer Byte to Memory	7-101
TIB	Transfer I to Byte	7-102
TJB	Transfer J to Byte	7-102
TKB	Transfer K to Byte	7-102
TEB	Transfer E to Byte	7-102
TTB	Transfer T to Byte	7-102
TOB	Transfer Operand to Byte	7-102
TMB	Transfer Memory to Byte	7-103
XOB	eXclusive-or Operand with Byte	7-103

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
45.*+X:a	Add Memory to Byte	A,C	AMB



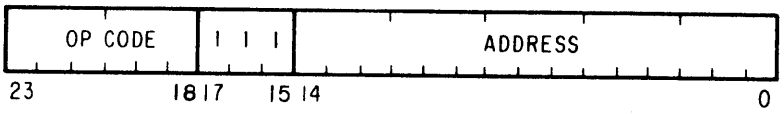
Bits 0-7 of the contents of the effective memory address are algebraically added to the contents of register B (A_0-A_7). Bits 8-23 of register A are unchanged.

0012.o	Add Operand to Byte	A,C	AOB



The 8-bit signed operand is algebraically added to the contents of the B register. (A_0-A_7). Bits 8-23 of register A are unchanged.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
60.7:a	Branch when Byte Address +1 in I ≠ 0	I	BBI
61.7:a	Branch when Byte Address +1 in J ≠ 0	J	BBJ



The contents of bits 22 and 23 of the specified index register (I or J) is incremented by one. If the result of this addition (in bits 22 and 23) is not 00_2, then the contents of register P (current PROGRAM ADDRESS) is replaced by the 15-bit effective memory address. If the result of the addition to bits 22 and 23 is 00_2, then bits 22 and 23 are set to 01_2 and bits 0-21 are incremented by one. If the resultant sum in bits 0-21 is zero, then register P advances to the next sequential program location and the index register is set to 40000000_8. Otherwise, the contents of register P are replaced by the 15-bit effective memory address.

In general, the BBI and BBJ instructions are used as special index register increments in order to sequentially reference consecutive bytes in memory via the EMB and RBM instructions. Consider the following example which will move 11 consecutive bytes starting from the third byte at location '200 to the first byte at location '300.

Example:

```
TMJ   = '60000200
TMI   = '20000300
TNK   11
EMB   0
RBM   0
BBI   *+1
BBJ   *+1
BWK   *-4
```

Occasionally, it is possible to use the address of a portion of index register I or J as a byte counter as well as a word pointer. This may be illustrated by the following example which will set the buffer starting at byte 3 of location '100 through byte 3 of location '102 to blanks.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC

Example:

```
TOB     "⅛"
TMI     = '77777775  bits 22 and 23 = 3, bits 0-21 = -3
RBM     '100+3
BBI     *-1
```

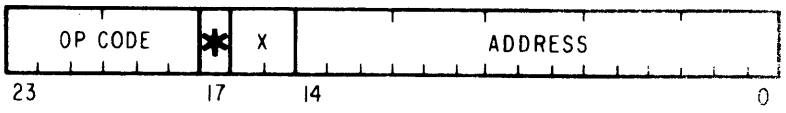
However, it should be noted this technique of using the index register as both a byte counter and word pointer may be used only in certain instances. Specifically, when the following relationship is true.

$$R\left(\frac{4\text{-b.n.}}{3}\right) = R\left(\frac{CT}{3}\right)$$

where:

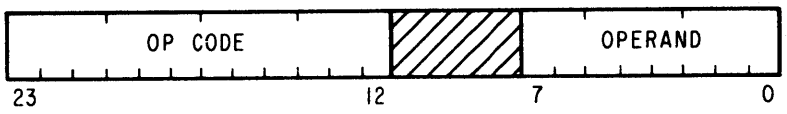
R ( ) = remainder
b.n. = the starting byte number (1, 2, or 3)
CT = the number of bytes to be referenced.

34.*+X:a — Compare Memory and Byte — C — CMB



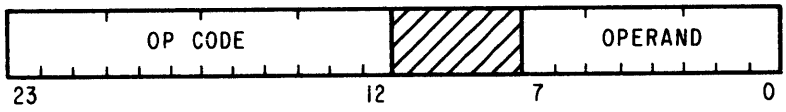
The contents of register B (A_0-A_7) and the contents of the effective memory address (M_0-M_7) are algebraically compared and the Condition register is set to the status of the result.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0014:o	Compare Operand and Byte	C	COB



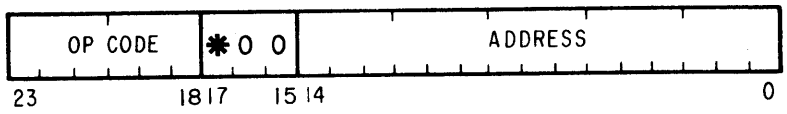
The 8-bit signed operand and the contents of register B (A_0-A_7) are algebraically compared and the Condition register is set to the status of the result.

0016:o	Dot Operand with Byte	A,C	DOB



A logical AND is performed between the 8-bit operand and the contents of register B (A_0-A_7). Bits A_8-A23 are unchanged.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
31.*+0:a	Extract Memory Byte	B,C	EMB



The effective memory address is added to the contents of register J, producing the word address which contains the byte to be extracted. The selected byte, as determined by the contents of bits 22 and 23 of index register J, is then placed in register B. The following table shows the correspondence between bits 22 and 23 of J and the byte to be extracted:

Bits 22 and 23 J Register	Byte Selection
01	Leftmost byte (bits 16-23 of EMA+J)
10	Middle byte (bits 8-15 of EMA+J)
11	Rightmost byte (bits 0-7 of EMA+J)
00	Rightmost byte (bits 0-7 of EMA+J)

The final address of any indirect sequence should not be indexed since implied indexing on register J takes place. If indexing is specified on the final address, then the specified index register will be ORed with register J prior to the add function with the EMA.

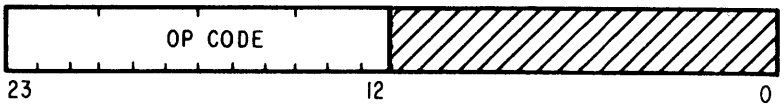
Example:

If J = '40000030
and K = '00000010 when the following is executed:

```
        EMB*   '40
'40     DAC*   '50,K
'42     DATA   "XYZ"
'60     DAC    '12
```

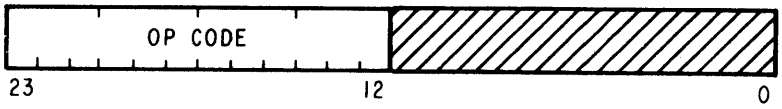
Then the character Y will be placed in register B. Note that the effective address of the indirect/index sequence is '12. However, '12 plus bits 0-15 of index register J ('30) yields the final address of '42. Since a byte specification of 10_2 was made in bits 22-23 of index register J, then the second byte (bits 8-15) of memory location '42 is placed in register B.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0010.	Extend Sign of Byte	A,C	ESB



The state of the register B sign bit (A_7) is copied into bit positions A_8-A_{23}, forming a sign extension of the byte.

0007.	Extend Zeros from Byte	A	EZB

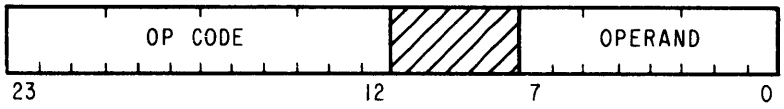


Bit positions A_8-A_{23} are set to ZERO. The contents of register B (A_0-A_7) are not affected.

NOTE

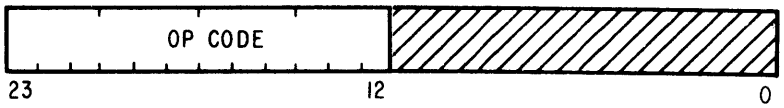
The Condition register is not affected.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0015:o	Kompare Operand and Byte	C	KOB



The 8-bit operand and the contents of register B (A_0-A_7) are logically compared and the Condition register is set according to the result.

0005.	Negate of Byte to Byte	A,C	NBB

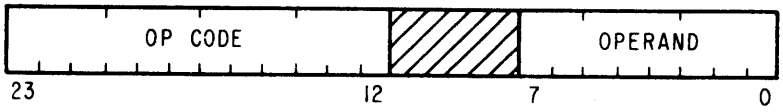


The contents of register B (A_0-A_7) are two's complemented. Bit positions A_8-A_{23} are unchanged.

NOTE

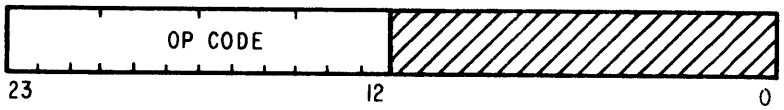
An Overflow will result when negating 2^7 (full-scale negative byte).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0004:o	Or Operand with Byte	A,C	OOB



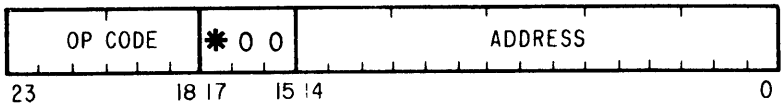
A logical OR is performed between the 8-bit operand and the contents of register B (A_0-A_7). Bits A_8-A_{23} are unchanged.

0006.	Positive of Byte to Byte	A,C	PBB



The absolute value of the contents of register B (A_0-A_7) is placed in register B.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
27.*+0:a	Replace Byte in Memory	M	RBM

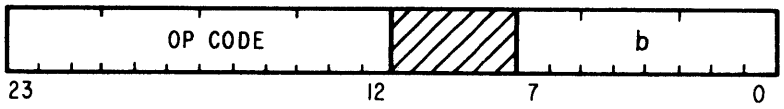


The effective memory address is added to the contents of register I producing the word address which contains the byte to be replaced. The selected byte, as determined by the contents of bits 22 and 23 of index register I, is then replaced by the contents of register B. The following table shows the correspondence between bits 22 and 23 of I and the byte to be replaced.

Bits 23 and 22 I Register	Byte Selection
01	Leftmost byte (bits 16-23 of EMA+I)
10	Middle byte (bits 8-15 of EMA+I)
11	Rightmost byte (bits 0-7 of EMA+I)
00	Causes no operation.

The final address of any indirect/index sequence should not be indexed since implied indexing on register I takes place. If indexing is specified on the final address, then the specified index register will be logically ORed with register I prior to the add function with the EMA.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0011:b	Query Bits of Byte	C	QBB

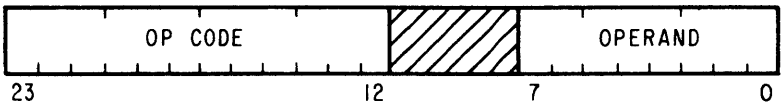


A logical AND is performed between operand bits 0-7 and the contents of register B. The Condition register is set according to the status of the result; i.e., positive, negative, or zero.

Examples:

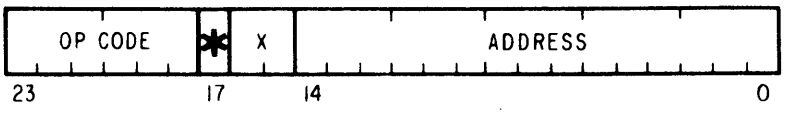
```
1.      TOA     B7      A = '00000200   C = Positive
        ...
        ...
        QBB     B7                      C = Negative
2.      TOA     B6      A = '00000100   C = Positive
        ...
        ...
        QBB     B6                      C = Positive
3.      TNA     1       A = '77777777   C = Negative
        ...
        ...
        DMA     MASK    A = '40000000   C = Negative
        ...
        ...
MASK    DATA    '40000000
```

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0013:o	Subtract Operand from Byte	A,C	SOB



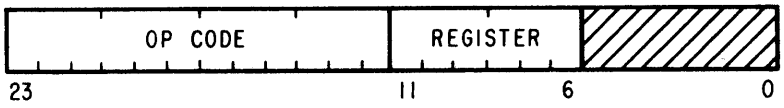
The 8-bit signed operand is algebraically subtracted from the contents of register B (A_0-A_7). Bits A_8-A_{23} are unaffected.

17.*+X:a	Transfer Byte to Memory	M	TBM



The contents of register B (A_0-A_7) replace the 8 least significant bits (0-7) of the contents of the effective memory address. Bits 8-23 of the memory word are unaffected.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0002.01	Transfer I to Byte	A	TIB
0002.02	Transfer J to Byte	A	TJB
0002.04	Transfer K to Byte	A	TKB
0002.10	Transfer E to Byte	A	TEB
0002.40	Transfer T to Byte	A	TTB

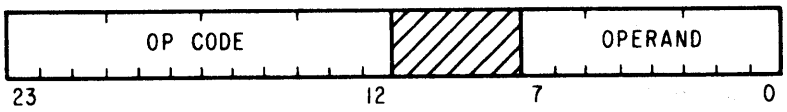


The least significant 8 bits (0-7) of the contents of the specified register replace the previous contents of register B (A_0-A_7). Bits A_8-A_{23} are unchanged.

NOTE

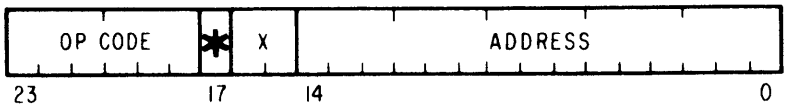
The Condition register is not affected.

0003:o Transfer Operand to Byte A,C TOB



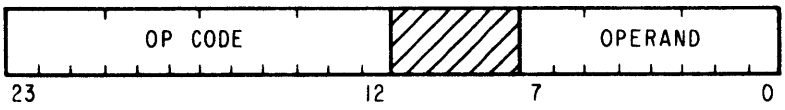
The 8-bit operand replaces the previous contents of register B (A_0-A_7). Bits A_8-A_{23} are unaffected.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
07.*+X:a	Transfer Memory to Byte	A,C	TMB



The 8 least significant bits (0-7) of the contents of the effective memory address replace the previous contents of register B (A_0-A_7). Bits A_8-A_{23} are unaffected.

0017:o	eXclusive-or Operand with Byte	A,C	XOB

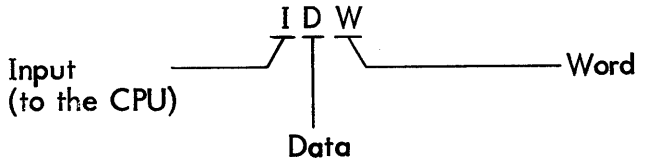


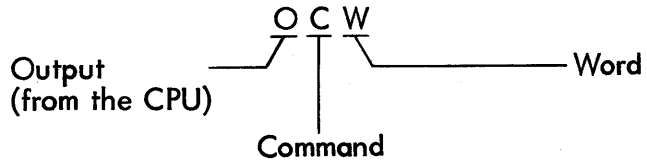
An Exclusive-or Operation is performed between the 8-bit operand and the contents of register B (A_0-A_7). Bits A_8-A_{23} are unchanged.

7-12 INPUT/OUTPUT INSTRUCTIONS

The Input/Output I/O instructions provide the required control for all communications between the CPU and the input/output structure. In addition to controlling data transfers between the CPU and peripheral units, the I/O instruction repertoire allows peripheral unit command functions and status testing to be placed under program control.

The specific I/O operation can be identified by examination of the individual instruction mnemonics. All I/O instruction mnemonics use the letter "W" to indicate that a full word is to be transferred between the CPU and the I/O structure. The first letter of the mnemonic indicates the direction of the transfer (input or output). The second letter indicates the type of word to be transferred. For example:





There is no "I/O hold", or delay, imposed by the hardware. All I/O instructions are executed unconditionally, i.e., the CPU is not forced to wait for a response from the I/O structure in order to complete the instruction execution cycle.

Although there is no built-in hold/delay provision, a programmed delay can be implemented if desired. At the beginning of each I/O instruction cycle, the Condition register is cleared. At the end of the execution phase of each I/O instruction, bit 2 (ZERO) is set to "ZERO" if the selected Channel was ready and accepted the command. If the selected channel was not ready, bit 2 of the Condition register remains set to "NOT ZERO". The program can test the "NOT ZERO" state of bit 2 with a branch instruction following the I/O instruction. When bit 2 is set to "NOT ZERO+, a programmed delay is implemented. For example:

```
ODW     '0103     Output word to Channel 1, Unit 3
BNZ     *-1       Delay if not ready
---               Continue if ready
```

An example of a channel being not ready is when the peripheral unit's data transfer capability is slower than that of the program loop and therefore cannot accept data as it is available from the channel. Another example occurs in a channel/multiunit environment where the channel is connected to peripheral unit A and peripheral unit B is selected for a data transfer.

In this instance, the channel remains not ready until a disconnect/connect sequence is performed and peripheral unit B is connected to the channel. Two cycles are required for the disconnect/connect sequence.

NOTE

Status returned to the Condition register immediately after completion of an I/O instruction refers to channel status only. A ready (" ZERO") condition indicates the channel accepted the I/O command. This does not imply the I/O operation was completed with the selected peripheral unit.

If the program selects a non-existent channel or unit, the channel accepts the command or data and leaves bit 2 of the Condition register set to " NOT ZERO" to indicate not ready. The channel will remain not ready for any subsequent commands.

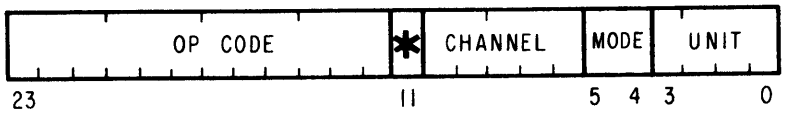
If the system is equipped with the Program Restrict/Instruction Trap option, all I/O instructions will be affected.

The I/O command modes are determined by the configuration of bits 5 and 4 of the OCW instruction and are as follows:

1. Normal - The Normal Channel Operation command is raised by bits 5 and 4 of the OCW being ZEROs (0,0).
2. Multiplex - This command is raised by bits 5 and 4 of the OCW being in a ZERO, ONE (0,1) configuration. (The CPU releases the channel to a master/slave pair of peripheral units.) (An XBC or IBC channel will not respond to a Multiplex command.)
3. Offline - This command is the same as the Multiplex command, except the I/O drivers in the channel are turned off, allowing the second CPU to share peripherals without need of peripheral switches. (Assumes control of I/O bus). The command is raised by bits 5 and 4 being in a ONE, ZERO (1,0) configuration.
4. Reset - This command operates the same as a Normal command, but resets the channel out of either the Multiplex or Offline mode. (Channel restored on-line, unit selected). This command is raised by bits 5 and 4 being in a ONE, ONE (1,1) configuration.

The following instructions are included in the Input/Output group.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0070.*+C.U	Output Command Word	C	OCW



An 8-bit or a 24-bit command word is transferred from the A register to the specified channel/unit combination and the Condition register is set to " ZERO" . If the selected channel is not ready, the Condition register remains set to " NOT ZERO" which allows a programmed delay if desired.

Bits 0-3 of the OCW instruction form a 4-bit paralleled unit code that is used to select a particular peripheral unit. The configuration of bits 4 and 5 determines the Multiplex or Offline mode (explained in paragraph 7-12) for a particular channel. The configuration of bits 6-10 det determines which channel is to be selected. Bit 11 is the Override Bit, and bits 12-23 define the general process that is to be performed.

If the Override bit (*) is set (ONE), the command word assumes immediate control over the channel. The contents of the A register are transferred to the channel and a disconnect/connect sequence is initiated. The condition register is set to " ZERO" to indicate the channel has accepted but not necessarily executed the command. Upon completion of the disconnect/connect sequence, the channel transfers the command word to the unit.

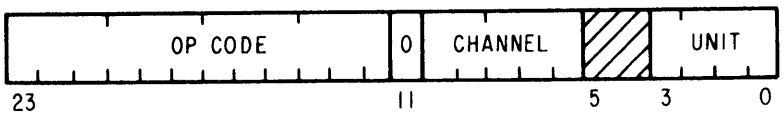
If the Override bit is not set (ZERO) and the OCW specifies a unit other than the unit connected to the channel and the channel is ready, the command word is accepted by the channel. The Condition register is set to " NOT ZERO" to indicate the channel is not ready. A disconnect/connect sequence is performed and the command is transferred to the unit. The Condition register is reset to " ZERO" to indicate ready.

NOTE

Following the execution of an OCW, the channel remains not ready until the peripheral unit accepts the data.

If the selected channel is a UBC channel and is actively engaged in a block transfer, executing an OCW with the Override bit set terminates the transfer sequence leaving the contents of the TAR/PAR and WCR intact. If the Override bit is not set and the UBC channel is engaged in a block transfer, the OCW instruction will be ignored. The Condition register will remain set to "NOT ZERO". Once a UBC channel is activated it will not accept an OCW with the Override bit not set until the word count is complete; i.e., all words in the block have been transferred and WCR equals zero.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0073.00+C.U	Input Status Word	A,C	ISW



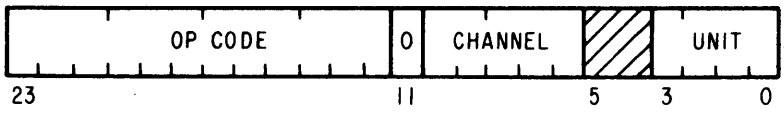
A status word is transferred from the specified channel/unit combination to register A and the Condition register is set to " ZERO" .

If the addressed channel/unit combination is not ready (see following notes) or status word is not available, the Condition register is set to " Not Zero" to allow a programmed delay.

NOTES

1. If the selected channel is in the process of executing a command (resulting from a previous OCW), the channel indicates not ready (Condition register remains set to " NOT ZERO") and ignores the ISW instruction until the peripheral unit accepts the OCW command. The channel indicates ready (Condition register set to " ZERO") and accepts the ISW when it is executed again.
2. If the ISW specifies a unit other than the unit connected to the channel, the channel indicates not ready and ignores the command. A disconnect/connect is initiated.
3. If the selected channel is a UBC channel engaged in a block transfer, the Condition register is set to " ZERO" and a 24-bit status word is transferred to the A register. Bits 0 through 7 contain the unit status and bit 23 contains the UBC busy stauts.
4. If the selected unit is receiving data as the result of an ODW instruction, the ISW will be accepted and the Condition register is set to " ZERO" .

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0071.00+C.U	Output Data Word	C	ODW



A data word is transferred from register A to the specified channel/unit combination and the Condition register is set to " Zero" .

If the channel is busy and cannot accept the data word, the Condition register is set to " Not Zero" to allow a programmed delay.

NOTES

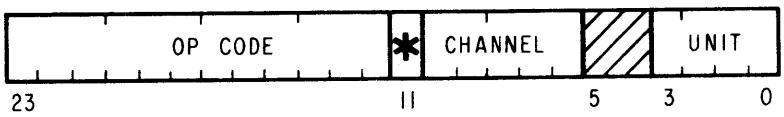
1. Although, a 24-bit word is transferred to the channel, the peripheral unit accepts only a predetermined number of bits (dictated by peripheral unit design).

2. For character-oriented units and units accepting data words of less than 24 bits, the data for transfer must be right-justified in the A register prior to executing the ODW instruction.

3. If ODW instruction specifies a unit other than the unit connected to the channel and the channel is ready, the channel accepts the ODW, sets the' Condition register to " ZERO" , and initiates a disconnect/connect sequence. After completion of the disconnect/connect sequence, the ODW is transferred to the unit. The channel indicates ready to subsequent I/O instructions.

4. If the ODW instruction specifies a UBC channel that is engaged in a block transfer, the Condition register remains set to " NOT ZERO" and the ODW is ignored. A UBC channel, once activated, will not accept an ODW instruction until the word count is complete, i.e., all words in the block have been transferred and WCR equals zero.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0072.*+C.U	Input Data Word	A,C	IDW



A data word is transferred from the specified channel/unit combination to register A and the Condition register is set to " Zero" .

If the channel is not ready or data from the specified unit is not available, the Condition register is set to " Not Zero" to allow a programmed delay.

NOTES

1. If the selected unit is in the process of executing a command as the result of a previous OCW instruction, the channel indicates not ready (Condition register remains set to " NOT ZERO") and the IDW is ignored. At the completion of the OCW, the Condition register is set to " ZERO" and the IDW instruction will be accepted by the channel.

2. If the selected unit is in the process of receiving data as a result of an ODW instruction and data is available from the unit, an ODW will be accepted and the Condition register set to " ZERO" .

3. If the IDW instruction specifies a unit other than the unit connected to the channel, the channel indicates not ready (Condition register remains set to " NOT ZERO"), ignores the instruction, and initiates a disconnect/connect sequence.

4. If a IDW instruction specifies a UBC channel that is engaged in a block transfer, the Condition register remains set to " NOT ZERO" (channel not ready) and the instruction is ignored. A UBC channel, once activated, will not accept an IDW instruction until the word count is complete; i.e., all words in the block have been transferred and WCR equals zero.

5. When a UBC channel is employed in a single-word programmed data transfer, an IDW instruction returns a "NOT READY" (C Register = NOT ZERO) condition if the channel is currently processing an output command. This situation is in effect regardless of the status of the input data from the peripheral unit.

6. If the Merge bit is a ONE, an OR is performed between the previous contents of register A and the incoming data word. This feature, in conjunction with a shift operation, allows input data characters to be packed in register A.

7. If the Merge bit (*) is ZERO, register A is cleared prior to the data transfer. Input data is right-justified in register A.

Example: Two 12-bit data characters are to be packed in register A.

```
IDW     '0102    Clear A and load first character from
                 channel 01, Unit 02.
BNZ     *-1      Wait if busy
LLA     12       Shift the contents of A left 12 bits
IDW*    '0102    Merge second character
BNZ     *-1      Wait if busy
...              Continue
```

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0071.40+C.U	Output Address Word	C	OAW

The unit is addressed in XBC channel (bits 0-2) and IBC channels (bits 0, 1) only.

The contents of register A are transferred to an appropriate register in the specified channel, or unit in XBC channel executions. The condition register is set to " Zero" .

NOTE

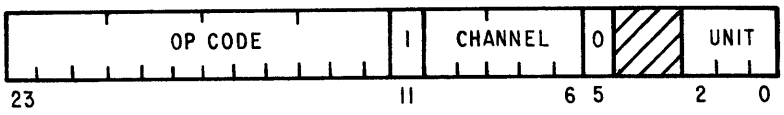
A UBC channel will always indicate ready for an OAW instruction. However, if the OAW specifies an invalid channel number, it will receive a "not ready" indication and the Condition register remains set to " Not Zero" . Since XBC/IBC channels involve a unit address, the unit must be " connected" before the instruction can be executed.

The OAW instruction does not activate a block-transfer channel. It transfers the starting memory address of the first of two parameter words from the A register to the TAR or PAR in the selected channel. In XBC channel operations the OAW instruction transfers the contents of register A to the unit; the channel has no register dedicated to this function.

If an OAW instruction addresses a UBC channel during a block transfer sequence, the sequence will be terminated.

If the OAW instruction addresses a PIOC, the Condition register remains set to " NOT ZERO": the instruction is executed automatically. In this instruction the four least significant bits (0-3) of the A register are transferred to the Interrupt Generator logic. These bits (unitarily) control the triggering of the one-to-four 1 microsecond interrupt pulses.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0073.40+C.O.U	Input Address Word	A,C	IAW



The current contents of the Transfer Address Register (TAR) in the specified channel are transferred to register A and the Condition register is set to "Zero".

The unit is addressed in IBC channel only.

NOTES

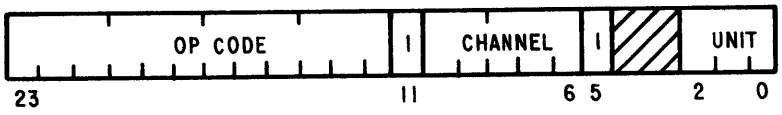
Bit 5 at the ZERO level distinguishes between the IAW and IPW instructions.

The UBC channel always indicate ready to an IAW instruction. The IBC channel must go to "Not Busy" before executing the instruction.

If the IAW instruction specifies an invalid channel or a standard channel, the Condition register remains set to "Not Zero" indicating channel not ready.

Refer to Section IV for a description of a block-transfer operation.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0073.40+C.4.U	Input Parameter Word	A,C	IPW



The current contents of the Parameter Address register (PAR) in the specified UBC channel are transferred to register A and the Condition register is set to " Zero" . (IPW instructions addressed to an IBC channel must specify via the unit address which of three-possible channel PARs is read.)

The unit is addressed in IBC channel only.

NOTES

1. Bit 5 at the ONE level distinguishes between the IPW and IAW instructions.

2. UBC channels always indicate ready to an IPW instruction. The IBC channel must go to " Not Busy" before executing the instruction.

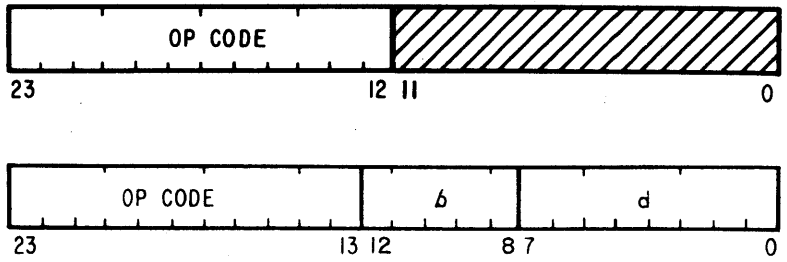
3. If the IPW instruction specifies an invalid channel or a standard channel, the Condition register remains set to " Not Zero" , indicating channel not ready.

4. Refer to Section IV for a description of a block-transfer operation.

7-13 BIT PROCESSOR INSTRUCTIONS

The bit (Boolean function) processor group of instructions include branches, logical manipulation, and interrogation of a specified bit selected from an effective memory address or the H register. In most instances, bit 2 (ZERO/NOT ZERO) of the Condition register is used to display either the result of an operation or the status of a bit before the operation is performed.

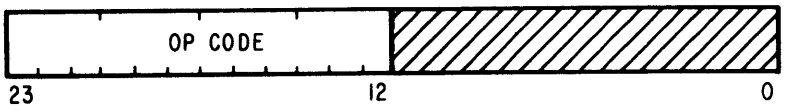
The bit processor employs two instruction word formats. The first format uses an (bits 12-23) OP CODE to specify the operation to be performed. The remaining 12 bits (bits 0-11) are undefined. The second instruction format contains a displacement, bit specification and an Op Code. Eight bits (bits 0-7) are added to the base address contained in register V to obtain a displacement from the base address which is an effective memory address for the word containing the bit in question. Five bits (bits 8-12) are used to select a specific bit in the effective memory address for an operation as specified in the 11-bit (bits 13-23) Op Code. Both instructions word formats are illustrated below.



The following instructions are included in the Bit Processor group.

MNEMONIC	INSTRUCTION	PAGE
DMH	Dot Memory with H	7-120
DNH	Dot Not (memory) with H	7-120
FBM	Flag Bit of Memory	7-124
NHH	Negate of H to H	7-119
OMH	Or Memory with H	7-121
ONH	Or Not (memory) with H	7-121
QBH	Query bit of H	7-119
QBM	Query bit of Memory	7-123
TFH	Transfer Flag to H	7-117
THM	Transfer H to Memory	7-124
TKV	Tranfer K to V	7-118
TMH	Transfer Memory to H	7-123
TVK	Transfer V to K	7-118
TZH	Transfer Zero to H	7-117
XMH	eXclusive-or Memory with H	7-122
XNH	eXclusive-or Not (memory) with H	7-122
ZBM	Zero Bit of Memory	7-125

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7742.	Transfer Zero to H	H,C	TZH

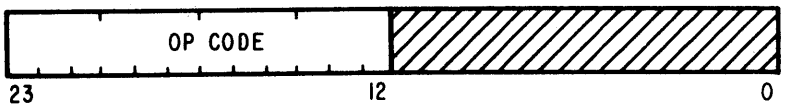


A ZERO is placed in register H. The Condition register is set to reflect the original contents of H.

NOTE

If the original contents of the H register was ZERO, Condition register Bit 2 is set to 1 (ZERO). If the contents was ONE, Bit 2 is set to 0 (NOT ZERO).

7743.	Transfer Flag to H	H,C	TFH

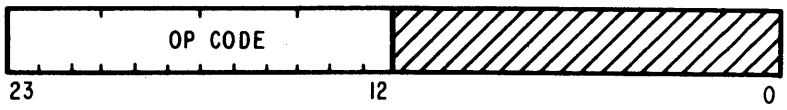


A ONE is placed in register H and the Condition register is set to reflect the original contents of H.

NOTE

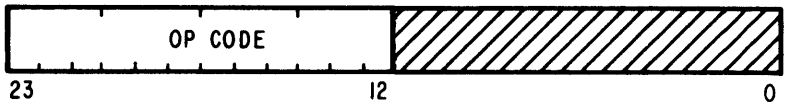
If the original contents of the H register was ZERO, Condition register Bit 2 is set to 1 (ZERO). If the contents was ONE, Bit 2 is set to 0 (NOT ZERO).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7744.	Transfer K to V	V	TKV



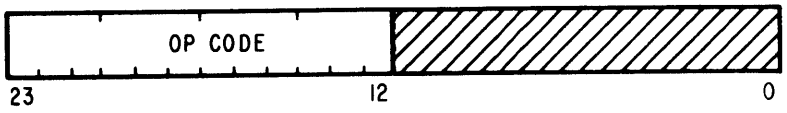
The 18 least significant bits of register K replace the present contents of register V. The Condition register is unaffected.

7745.	Transfer V to K	K	TVK



The contents of register V are transferred to the 18 least significant bit positions of register K. The Condition register is unaffected.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7746.	Query Bit of H	C	QBH

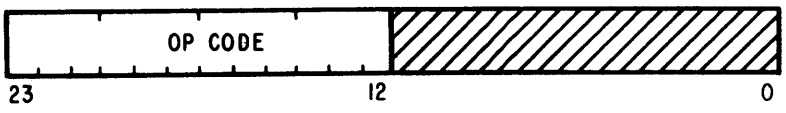


The H register bit is tested and the Condition register is set to display the result of the query.

NOTE

The Condition register is cleared. If the resultant content of the H register is ZERO, Condition register Bit 2 is set to 1 (ZERO). If the content is ONE, Bit 2 is set to 0 (NOT ZERO).

7747.	Negate of H to H	H,C	NHH

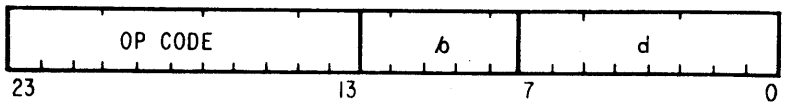


The current content of register H is complemented and returned to H. The Condition register is set to display the result.

NOTE

The Condition register is cleared. If the resultant content of the H register is ZERO, Condition register Bit 2 is set to 1 (ZERO). If the content is ONE, Bit 2 is set to 0 (NOT ZERO).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7750.*b*:d	Dot Memory with H	H,C	DMH

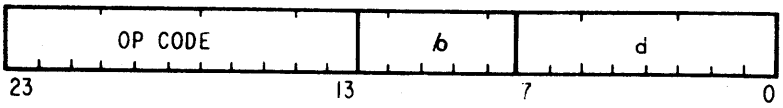


A logical AND is performed between the selected bit in the effective memory address and the contents of register H. The result is returned to the H register and the Condition register is set to display the result.

NOTE

The Condition register is cleared. If the resultant content of the H register is ZERO, Condition register Bit 2 is set to 1 (ZERO). If the content is ONE, Bit 2 is set to 0 (NOT ZERO).

7752.*b*:d	Dot Not (memory) with H	H,C	DNH

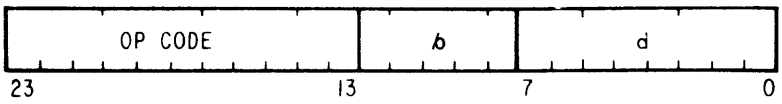


A logical AND is performed between the complement of the selected bit in the effective memory address and the content of register H. The result is returned to the H register and the Condition register is set to display the result.

NOTE

The Condition register is cleared. If the resultant content of the H register is ZERO, Condition register Bit 2 is set to 1 (ZERO). If the content is ONE, Bit 2 is set to 0 (NOT ZERO).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7754.*b*:d	Or Memory with H	H,C	OMH

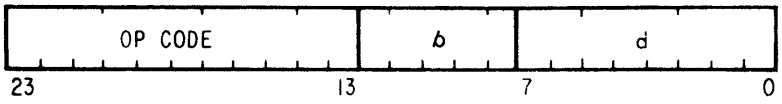


A logical OR is performed between the selected bit in the effective memory address and the content of register H. The Condition register is set to display the result.

NOTE

The Condition register is cleared. If the resultant content of the H register is ZERO, Condition register Bit 2 is set to 1 (ZERO). If the content is ONE, Bit 2 is set to 0 (NOT ZERO).

7756.*b*:d	Or Not (memory) with H	H,C	ONH

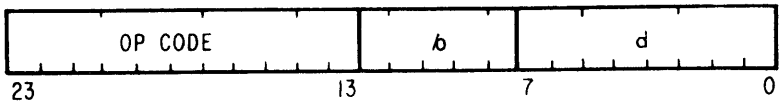


A logical OR is performed between the complement of the selected bit in the effective memory address and the content of register H. The Condition register is set to display the result.

NOTE

The Condition register cleared. If the resultant content of the H register is ZERO, Condition register Bit 2 is set to 1 (ZERO). If the content is ONE, Bit 2 is set to 0 (NOT ZERO).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7760.*b*:d	eXclusive-or Memory with H	H,C	XMH

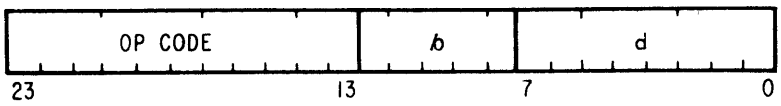


An EXCLUSIVE-OR function is performed between the selected bit in the effective memory address and the content of register H. The Condition register is set to display the result.

NOTE

The Condition register is cleared. If the resultant content of the H register is ZERO, Condition register Bit 2 is set to 1 (ZERO). If the content is ONE, Bit 2 is set to 0 (NOT ZERO).

7762.*b*:d	eXclusive-or Not (memory) with H	H,C	XNH

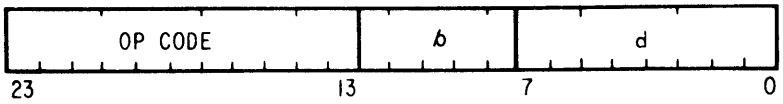


An EXCLUSIVE-OR function is performed between the complement of the selected bit in the effective memory address and the content of H register. The Condition register is set to display the result.

NOTE

The Condition register is cleared. If the resultant content of the H register is ZERO, Condition register Bit 2 is set to 1 (ZERO). If the content of ONE, Bit 2 is set to 0 (NOT ZERO).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7764.*b*:d	Transfer Memory to H	H,C	TMH

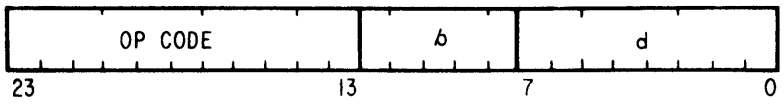


The selected bit in the effective memory address is transferred to register H. The Condition register is set to display the resultant content of the H register.

NOTE

The Condition register is cleared. If the resultant content of the H register is ZERO, Condition register Bit 2 is set to 1 (ZERO). If the resultant content is ONE, bit 2 is set to 0 (NOT ZERO).

7766.*b*:d	Query Bit of Memory	C	QBM

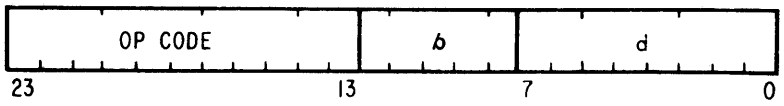


The selected bit in the effective memory address is tested and the Condition register is set to display the result of the query.

NOTE

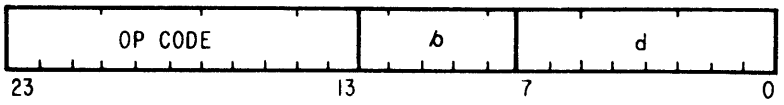
The Condition register is cleared. If the resultant content of the H register is ZERO, Condition register Bit 2 is set to 1 (ZERO). If the resultant content is ONE, Bit 2 is set to 0 (NOT ZERO).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7770.*b*:d	Transfer H to Memory	M	THM



The content of register H is placed in the selected bit position in the effective memory address. The Condition register is not affected.

7772.*b*:d	Flag Bit of Memory	M,C	FBM

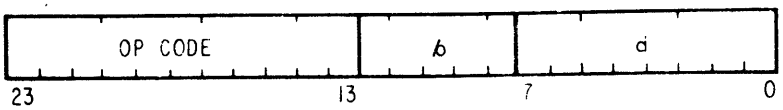


A ONE is placed in the selected bit position in the effective memory address. The Condition register is set to display the original state of the selected bit in memory.

NOTE

If the original state of the selected bit in memory was ZERO, Condition register Bit 2 is set to 1 (ZERO). If the original state was ONE, Bit 2 is set to 0 (NOT ZERO).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7774.*b*:d	Zero Bit of Memory	M,C	ZBM



A ZERO is transferred to the selected bit position in the effective memory address. The Condition register is set to display the original state of the selected bit in memory.

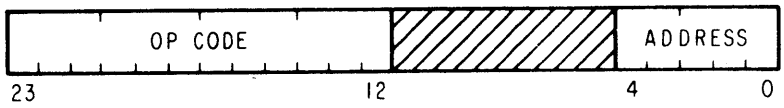
NOTE

If the original state of the selected bit in memory was ZERO, Condition register Bit 2 is set to 1 (ZERO). If the original state was ONE, Bit 2 is set to 0 (NOT ZERO).

7-14 PROGRAM RESTRICT INSTRUCTIONS

The following instructions provide control for the optional Program Restrict System (available only in a non-virtual memory SLASH 6).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0067:a	Branch and Link Unrestricted	J,P	BLU

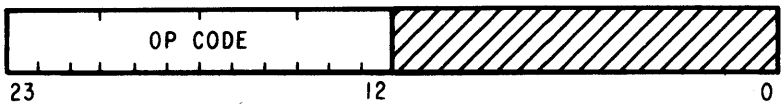


The next sequential address (PROGRAM ADDRESS +1) replaces the contents of register J and the contents of register P (current PROGRAM ADDRESS) are replaced by the 5-bit immediate memory address.

NOTES

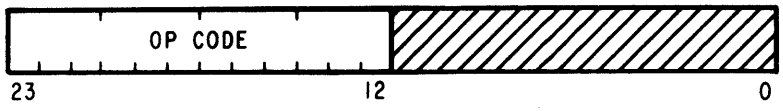
1. Execution of the BLU instruction will turn OFF the Program Restricted Flag (PRF). If the computer is in a HALT condition and thePRF is ON, the BLU instruction will be treated as a NOP instruction.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0056.	Transfer Double to Limit registers	LL,UL	TDL



The contents of bits E_0-E_{17} replace the previous contents of the Lower Limit (LL) register and the contents of bits A_0-A_{17} replace the previous contents of the Upper Limit (UL) register. Bits A_{21} and A_{22} set the restrict mode flags.

0057.	Transfer Limit registers to Double	E,A	TLD



The contents of the Limit registers replace the previous contents of register D (E and A). The Upper Limit register contents are transferred to bits A_0-A_{17} and the contents of the Lower Limit register are transferred to E_0-E_{17}. The states of the restrict mode flags are transferred to bits A_{21} and A_{22}. All other bits in E and A are reset to ZERO.

7-15 PRIORITY INTERRUPT CONTROL INSTRUCTIONS

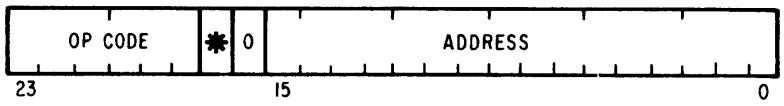
The Priority Interrupt instruction group provides the means for program control of external interrupts. External interrupts may be selectively armed, disarmed, enabled or inhibited under program control. Other instructions provide the means for holding and releasing external interrupts, while others are available for transferring control upon interrupt detection. For a detailed description of the SLASH 6 Priority Interrupt System, refer to Section V of this manual.

If the SLASH 6 system is equipped with the optional program restrict system and instruction Trap, the following Priority Interrupt instructions will be affected:

a) Hold eXternal Interrupts (HXI)
b) Release eXternal Interrupts (RXI)
c) Unitarily Arm group 1 interrupts (UA1)
d) Unitarily Disarm group 1 interrupts (UD1)
e) Unitarily Enable group 1 interrupts (UE1)
f) Unitarily Inhibit group1 interrupts (UI1)
g) Transfer Double to group 1 (TD1)
h) Transfer Double to group 1 (TD4)

The following instructions are included in the Priority Interrupt group:

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
25.*+0:A	Branch and Save return —— Long	P	BSL



The next sequential address (PROGRAM ADDRESS +1), along with the contents of the Condition register, are stored in the 16-bit effective memory address (EMA). The contents of register P (current PROGRAM ADDRESS) are then replaced by the address following the effective memory address (EMA +1).

This instruction is normally used to enter an interrupt subroutine because it provides a means of returning to the main program at the point of interrupt and saves the machine status (Condition) at the time of the interrupt.

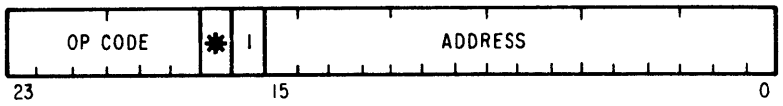
NOTES

1. The contents of the Condition register are stored in bit positions 16-19 of the EMA and the return address (PROGRAM ADDRESS +1) is stored in bits 0-15. The remaining bits are set to ZEROs, however, refer to next note 2 for variation on bit 20.

2. The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted.

3. External interrupts are prohibited for the period of one instruction following the execution of this instruction.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
25.*+2:A	Branch and Reset interrupt —— Long	C,P	BRL



The highest-level active interrupt is reset (i.e., returned to the inactive state) and the contents of register P (current PROGRAM ADDRESS) are replaced by the 16-bit effective memory address.

BRL is normally used to exit an interrupt subroutine. If BRL contains an indirect reference, the last word in the indirect address chain contains the previous status (i.e., C register contents at the time of the interrupt) in bit positions M_{16}-M_{19} and the return address in bit positions M_0-M_{15} as a result of the BSL instruction. The C register is restored and the program branches to the return address (restarting the machine to the preinterrupt status).

Example:

```
                .
                .
                TMA
L               AMA
                SMA             Interrupt occurs (EXM K).
                .
                .
K               BSL     M       Dedicated interrupt location.
M               ***             M becomes L+1 as a result of BSL at K.
                .               The C register contents are stored in
                .               M16-M19.
                BRL*            Restore C register and return to L+1.
```

NOTES

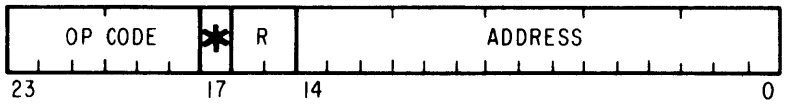
The BRL will not reset the interrupt if external interrupts have been held by an HXI instruction. Control will be returned to the effective memory address.

Those executive traps, which are not affected by the HXI instruction, will be reset by the BRL.

The immediate memory reference cannot be indexed; however, indexing indirect references is permitted, e.g.,

```
                BRL*    X
                .
                .
X               DAC     Y,K
```

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
27.*+1:a	Hold interrupts and Transfer I to memory	M	HTI
27.*+2:a	Hold interrupts and Transfer J to memory	M	HTJ
27.*+3:a	Hold interrupts and Transfer K to memory	M	HTK



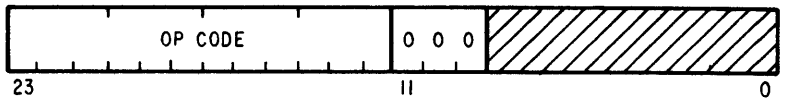
The contents of register I, J or K replace the previous contents of the effective memory address and external interrupts are prohibited for the period of one instruction following the execution of this instruction.

NOTE

The immediate memory reference cannot be indexed; however, indexing of indirect references is permitted, e.g.,

```
          HTI*      M
           .
           .
M         DAC       A,K
```

0066.0	Hold eXternal Interrupts	None	HXI

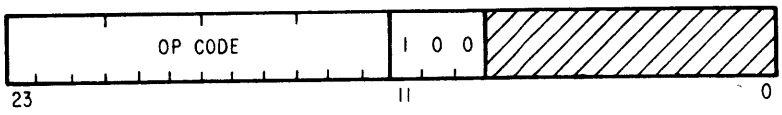


The activation of any external interrupt is prohibited. The prohibition is effective immediately upon execution of the instruction and lasts until the interrupts are released (see RXI instruction). Executive traps (Group 0, Levels 5-7) are prohibited from becoming active while the HXI is in effect.

NOTE

Only the three executive traps mentioned are affected by this instruction.

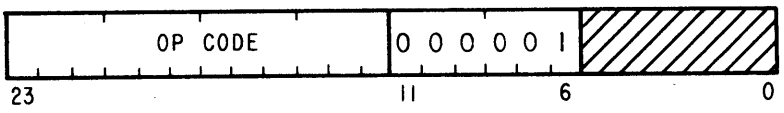
INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0066.4	Release eXternal Interrupts	None	RXI



The prohibition imposed by the HXI instruction is removed, allowing any external interrupt to be activated 1 cycle after this instruction. This permits the next sequential instruction to be executed without external interruption.

If any of the affected executive traps have been triggered while an HXI was in effect, the highest level will come in first after the RXI instruction.

0064.01	Transfer Double to group 1	1 A/D, 1 E/I	TD1

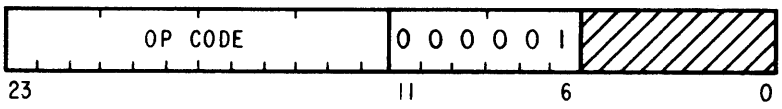


The contents of register D (E and A) replace the previous contents of the Arm/Disarm (A/D) and Enable/Inhibit (E/I) registers of interrupt group 1. The contents of E are transferred to the A/D register and the contents of A are transferred to the E/I register.

NOTE

The external interrupt structure is cleared by the execution of this instruction.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0065.01	Transfer group 1 to Double	E,A	T1D

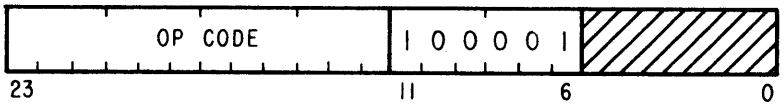


The contents of the Arm/Disarm (A/D) and Enable/Inhibit (E/I) registers of interrupt group 1 replace the previous contents of register D (E and A). The contents of the A/D register are transferred to register E and the contents of the E/I register are transferred to register A.

NOTE

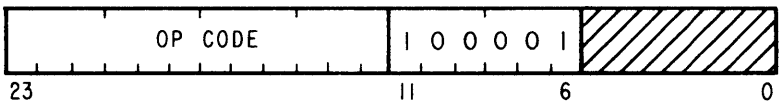
The states of the external interrupts are not affected by the execution of this instruction.

0065.41	Transfer group 1 to Double	E,A	T4D



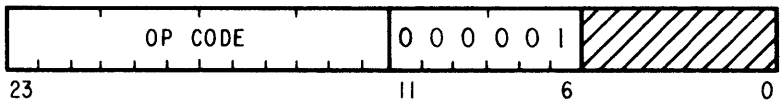
The contents of the request and active registers of interrupt group 1 replace the previous contents of register D (E and A). The contents of the request register are transferred to E, and the contents of the active register are transferred to A.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0064.41	Transfer Double to group 1	1 Request, Active	TD4



If armed, the contents of register D (E and A) are ORed with the current contents of the request and active registers of interrupt group 1. The contents of E are ORed with the request register and the contents of A are ORed with the active register.

0060.01	Unitarily Arm group 1 interrupts	1 A/D	UA1



Any number of the 24 interrupt levels in group 1 are selectively armed; i.e., the selected bit(s) of the Arm/Disarm (A/D) register are set to ONE.

The corresponding bit(s) of register A must be set to select the appropriate level(s) prior to executing this instruction.

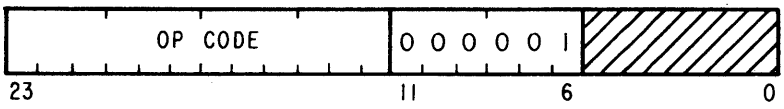
Example: Arm levels 1 and 3, group 1

```
TOA     B1B3    Select levels 1 and 3
                (set bits 1 and 3 of A)
UA1             Arm selected levels of group 1
```

NOTE

Execution of this instruction does not clear the interrupt structure and, therefore, does not affect any interrupt levels other than those selected. If a level selected for arming is already armed, it is not cleared by the execution of this instruction.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0061.01	Unitarily Disarm group 1 interrupts	1 A/D	UD1



Any number of the 24 interrupts levels in group 1 are selectively disarmed i.e., the selected bits of the Arm/Disarm (A/D) register are reset to ZERO.

The corresponding bit(s) of register A must be set to select the appropriate level(s) prior to executing this instruction.

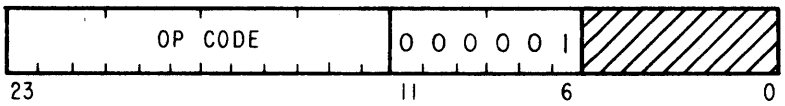
Example: Disarm level 2, group 1

```
TOA     B2      Select level 2 (set bit 2 of A)
UD1             Disarm selected level of group 1
```

NOTE

Execution of this instruction will clear only those levels which are selected. The remaining levels will not be affected.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0062.01	Unitarily Enable group 1 interrupts	1 E/I	UE1



Any number of the 24 interrupt levels in group 1 are selectively enabled, i.e., the selected bits of the Enable/Inhibit (E/I) register are set to ONE.

The corresponding bit(s) of register A must be set to select the appropriate level(s) prior to executing this instruction.

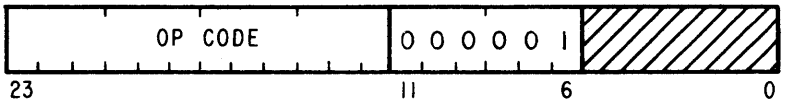
Example: Enable levels 0, 2 and 5, group 1

```
TOA     B0B2B5      Select levels 0, 2, 5
                    (set bits 0, 2 and 5 of A)
UE1                 Enable selected levels of group 1
```

NOTE

Execution of this instruction does not clear the interrupt structure and, therefore, does not affect any interrupt levels other than those selected. If a level selected for enabling is already enabled, it is not cleared by the execution of this instruction.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0063.01	Unitarily Inhibit group 1 interrupts	1 E/I	UI1



Any number of the 24 interrupt levels in group 1 are selectively inhibited; i.e., the selected bits of the Enable/Inhibit (E/I) register are reset to ZERO.

The corresponding bit(s) of register A must be set to select the appropriate level(s) prior to executing this instruction.

Example: Inhibit levels 1, 4 and 7 of group 1

```
TOA     B1B4B7      Select levels 1, 4, 7
                    (set bits 1, 4 and 7 of A)
UI1                 Inhibit selected levels of group 1
```

NOTE

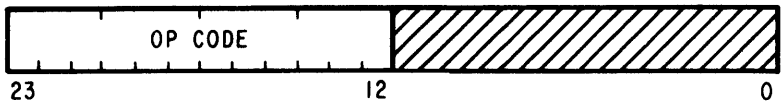
Execution of this instruction does not clear the interrupt structure and, therefore, does not affect any interrupt levels other than those selected. If one or more of the selected levels is active upon execution of this instruction, the level(s) will be placed in a "permissive" state.

7-16 MISCELLANEOUS INSTRUCTIONS

The following instructions are included in the Miscellaneous group because they do not fall into any defined functional group.

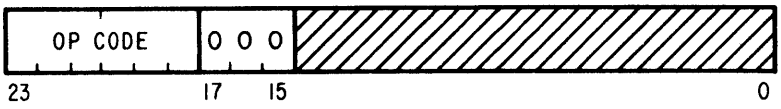
MNEMONIC	INSTRUCTION	PAGE
EXM	EXecute Memory	7-143
EZB	Extend Zeros from Byte	7-145
GAP	Generate Argument Pointer	7-141
HIT	Hold Interval Timer	7-146
HLT	Halt	7-140
NOP	No OPeration	7-140
QBB	Query Bits of Byte	7-144
QSS	Query Sense Switches	7-145
RCT	Release Clock Time	7-147
RPT	Release Processor Time	7-146
USP	Update Stack Pointer	7-142

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0000.	HaLT	P	HLT



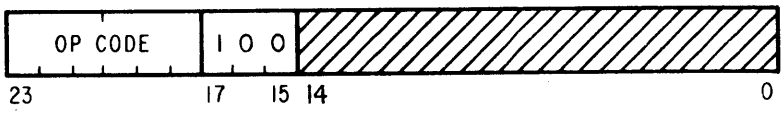
The PROGRAM ADDRESS (i.e., the contents of the P register) is advanced by one and program execution is terminated. When the RUN switch is depressed, execution will begin at the location defined by the PROGRAM ADDRESS.

62.0	No OPeration	P	NOP



The PROGRAM ADDRESS is advanced by one and program execution continues with the next instruction.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
24.4:o	Generate Argument Pointer	I,J	GAP



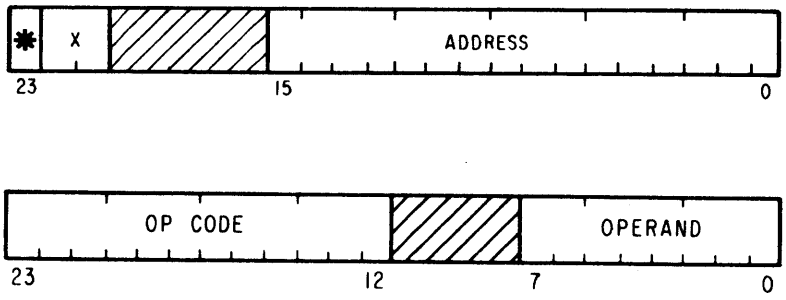
The contents of register J are assumed to be the first address in an indirect memory reference sequence. The effective memory address derived from this indirect sequence replaces the previous contents of register I. The contents of register J and the 15-bit operand are added, and the result is placed in register J.

The purpose of a GAP instruction is to generate an effective memory address which points to one or more data words not directly available to a subroutine. This is illustrated in the following example where subroutine B requires the data contained in location Y.

Example:

```
A       BLJ     B       (J) =  C, (P) = B
C       DAC*    X
D       ...             RETURN
        ...
X       DAC     Y
Y       DATA    2
        ...
B       GAP     1       (I) = Y, (J) = (J) +1
        TMA     0,I     (A) = 2
        BUC     0,J     (P) = D
```

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0055:o	Update Stack Pointer	K,C	USP



The contents of register K are replaced by the contents of the effective memory address. The 8-bit signed operand is then added to the contents of the effective memory address.

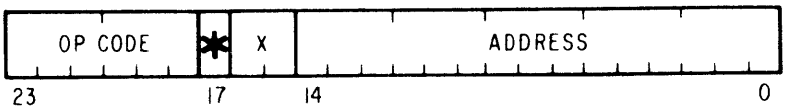
Example:

```
          BLJ     ENT     Call re-entrant routine
          ...
ENT       TRM*    SP      Save registers in stack
          USP     5       Update Stack Pointer (K) = stack,
                          (SP) = stack +5
          DAC     SP
          ...
          HTK     SP      Reset stack pointer
          TMR*    SP      Restore registers
          BUC     0,J     Return
SP        DAC     STACK   Stack pointer
STACK     BLOK    5N      Where N represents maximum number of
                          re-entrant levels
```

NOTES

1. The Condition register is set to reflect the result of the operand addition.
2. External interrupts are prohibited for the period of one instruction following this instruction.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
40.*+X:a	EXecute Memory	See Notes	EXM



The instruction located in the effective memory address is executed as though it were at the address of the EXM.

In the case that the referenced instruction is a two word instruction, the second word must follow the EXM.

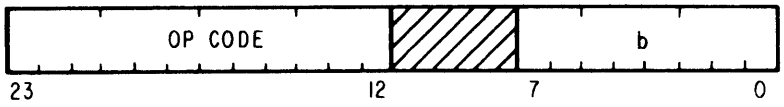
Example:

```
        EXM   M
        DAC   L        Second word
        ...
M       AOM   10       Two word instruction
        AOM   20
        AOM   30
```

NOTES

1. The registers affected will depend on the instruction in the effective memory address.
2. External interrupts are prohibited for the period of one instruction following the execution of this instruction.
3. The PROGRAM ADDRESS (Contents of P register) is not advanced when this instruction is executed.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0011:b	Query bits of Byte	C	QBB

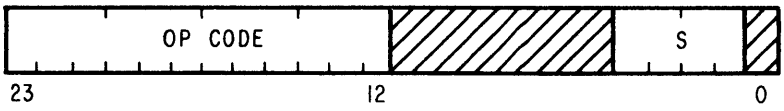


A logical AND is performed between operand bits 0-7 and the contents of register B. The Condition register is set according to the status of the result; i.e., positive, negative, or zero.

Examples:

```
          TOA     B7        A = '00000200    C = Positive
          ...
          ...
          QBB     B7                         C = Negative
          TOA     B6        A = '00000100    C = Positive
          ...
          ...
          QBB     B6                         C = Positive
          TNA     1         A = '77777777    C = Negative
          ...
          ...
          DMA     MASK      A = '40000000    C = Negative
          ...
          ...
MASK      DATA    '40000000
```

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0001:s	Query Sense Switches	C	QSS

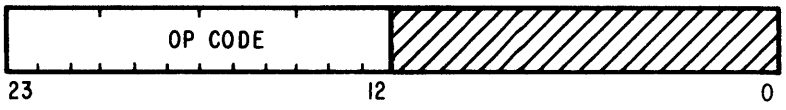


A logical AND is performed between operand bits 1-4 and the state(s) of the sense switches. The Condition register is set accordingly.

Example: Test to see if either SS2 or SS3 are on, or if both are on.

```
QSS     B2B3
```

0007.	Extend Zeros from Byte	A	EZB

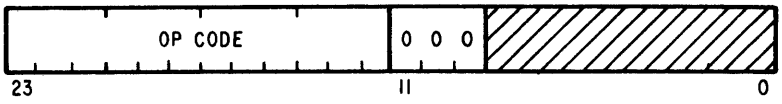


Bit positions A_8-A_{23} are set to ZERO. The contents of register B (A_0-A_7) are not affected.

NOTE

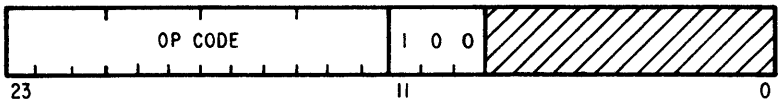
The Condition register is not affected.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0077.0	Hold Interval Timer	None	HIT



The CPU's Interval Timer is halted and will remain so until released by an RPT or RCT instruction.

0077.4	Release Processor Time	None	RPT



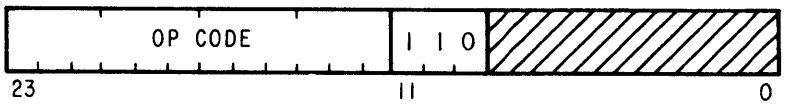
The CPU's Interval Timer is started; i.e., allowed to begin its counting sequence. The Processor Time mode allows the Interval Timer to count CPU time only. Counting is inhibited when an I/O block controller channel takes a memory cycle or when an interrupt is active.

Once started, the timer counts until held by a HIT instruction or until the CPU is halted. At each one microsecond interval, the contents of register T are decremented by one and tested for zero. If the contents of T are zero, an executive interrupt is triggered.

NOTE

The timer is not stopped by the execution interrupt.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
0077.6	Release Clock Time	None	RCT



The CPU's Interval Timer is started, i.e., allowed to begin its counting sequence. The Clock Time mode causes the Interval Timer to count continuously.

Once started, the timer will count until held by an HIT instruction. At each one microsecond interval, the contents of register T are decremented by one and tested for zero. If the contents of T are zero, an executive interrupt is triggered.

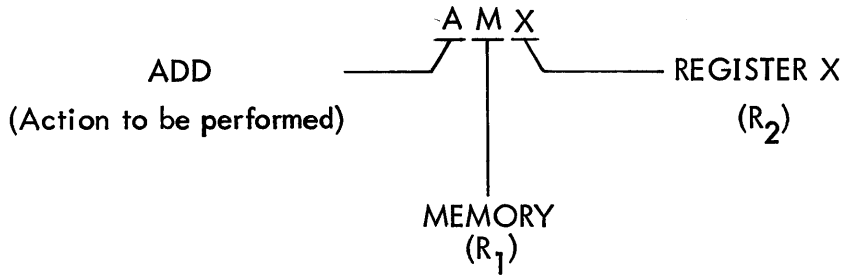
NOTE

The timer is not stopped by the interrupt.

7-17 SCIENTIFIC ARITHMETIC UNIT INSTRUCTIONS

The instruction set for the Scientific Arithmetic Unit is divided into five functional groups: arithmetic, transfer, branch, compare, and interrupt control. In the descriptions that follow, each instruction is defined and the time is given in cycles and concurrent cycles available. The concurrent cycles, if any, occur after the instruction has been initiated by the SAU. The SAU is designed to operate on normalized floating point numbers, and all descriptions of the arithmetic instructions are based on this fact. If an unnormalized operand is used in an arithmetic operation the results are not considered valid.

Standard arithmetic instructions - add, subtract, multiply, and divide - as well as square, square root, fix and float are included in the group. The instruction mnemonics provide a brief definition of specific operations to be performed. The first letter in the mnemonic specifies the action or type of operation that is to be performed. The second letter identifies the first quantity or reference (R_1) to be used in the operation and the third letter identifies the second reference (R_2). For example:



In the majority of SAU arithmetic instructions, the result of the operation remains in R_2 while R_1 remains unchanged (except where R_1 and R_2 are the same).

Unless otherwise noted, each arithmetic operation sets a bit in the SAU Condition (Y) register to reflect the status of the result. Various conditions are described below:

a) Positive - The result is arithmetically greater than zero, indicated by a ONE in bit position 3 of the Y register. A ZERO in bit position 3 indicates "Not Positive".

b) Zero -- All of the mantissa bits comprising the quantity under consideration are ZERO and the exponent is '201, indicated b a ONE in bit position 2 of the Y register. A ZERO in bit position 2 indicates "Not ZERO".

c) Negative - The result is arithmetically less than zero, indicated by a ONE in bit position 1 of the Y register. A ZERO in bit position ONE indicates "Not Negative".

d) Overflow - An overflow results from an arithmetic operation which causes exponent overflow, i.e., an exponent greater than $2^7 - 1$ (127) or less than -2^7 (-128).

NOTE

If the SAU Overflow/Underflow executive trap is enabled, any instruction causing the overflow bit of the Y register to be set will cause an interrupt.

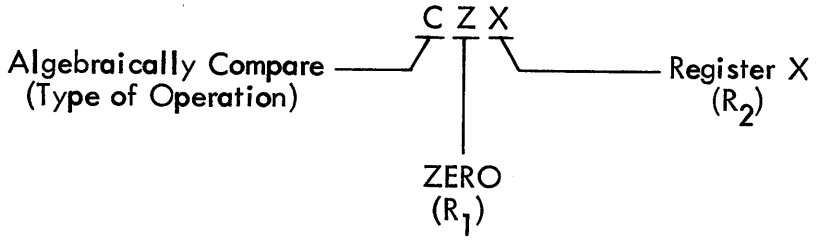
Bits 1, 2 and 3 (Negative, Zero, Positive) of the Y register are normally mutually exclusive. In certain instances it is desirable to know what operation caused an overflow, e.g., a division by zero. The following operations cause more than two bits to be set in the Y register:

Division by zero sets bits 0, 2, 3 ('15)

$\sqrt{-x}$ sets bits 0, 1, 2, 3 ('17)

Float to Fix, X 8388607 sets bits 0, 1, 3 ('13)

The algebraic compare instructions are included in the SAU instruction set which compare two referenced, signed (+ or -) quantities. The Y condition register is set according to the result of the comparison. Algebraic comparisons are identified by the letter "C" as the first letter in the instruction mnemonic (e.g., CZX). The second letter in the mnemonic code identifies the first of the compared quantities (R_1) and the remaining letter identifies the second quantity (R_2). For example:



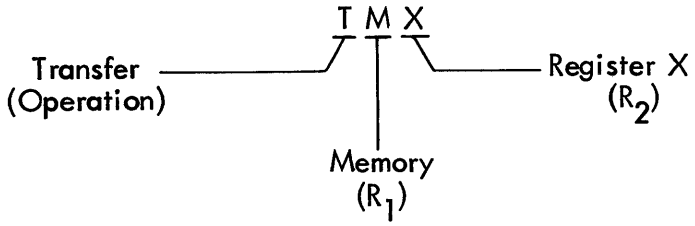
Comparisons are performed according to the following formula:

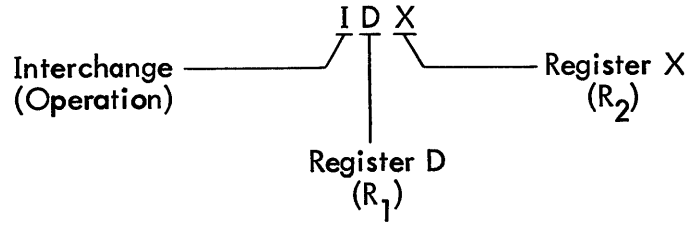
$$R_2 - R_1 = Y \text{ (positive, zero, or negative)}$$

Therefore, $R_2 > R_1$, $R_2 < R_1$, and $R_2 = R_1$, will set the condition (Y) register to positive (+) negative (-), and zero (0), respectively.

Two instructions provide control of the SAU interrupt. These instructions either release or hold the interrupt.

The transfer instruction group includes various types of operations. Among these are transfers between memory and registers, registers and memory, and register-to-register. The transfer operation mnemonic code describes the individual operation. What operation is to be performed is described by the first letter in the mnemonic; "T" for transfer and "I" for interchange. The second and third of the mnemonic specify the source (R_1) and destination (R_2) of the transfer, respectively. Listed on the following page are two examples:





With the exception of the interchange instruction, the transfer source (R_1) is not altered as a result of the execution of a transfer instruction.

The following instructions are included in the SAU group.

ARITHMETIC

Mnemonic	Instruction	Page
AAX	Add A register to X register	7-152
ADX	Add D register to X register	7-152
AMX	Add Memory to X register	7-152
AOW	Add Operand to W register	7-153
AOX	Add Operand to X register	7-153
DAX	Divide A register into X register	7-154
DDX	Divide D register into X register	7-154
DMX	Divide Memory into X register	7-155
DOX	Divide Operand into X register	7-155
FAX	Floating normalize of A register to X register	7-156
FXA	Fix of A register to A register	7-157
INX	INverse of X register	7-157
MAX	Multiply A register and X register	7-158
MDX	Multiply D register and X register	7-158
MMX	Multiply Memory and X register	7-159
MOX	Multiply Operand X register	7-159
NXX	Negative of X register to X register	7-160
PXX	Positive of X register to X register	7-160
SAX	Subtract A register from X register	7-161
SDX	Subtract D register from X register	7-161
SEX	SquarE X register	7-162
SMX	Subtract Memory from X register	7-162
SOX	Subtract Operand from X register	7-163
SRX	Square Root of X register	7-163

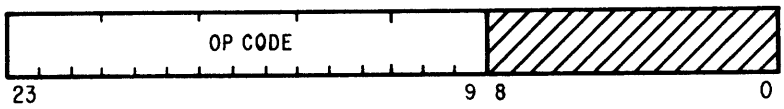
BRANCH

COMPARE

INTERRUPT

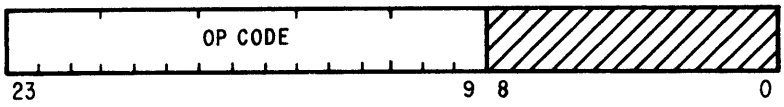
TRANSFER

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7707.0	Add A register to X register	X,Y	AAX



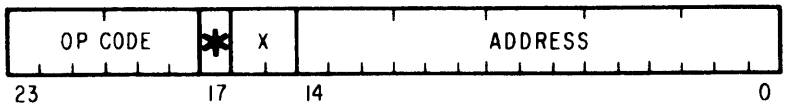
The signed integer in the A register is converted to floating-point format and added to the number in the X register. The sum replaces the previous contents of the X register.

7710.0	Add D register to X register	X,Y	ADX



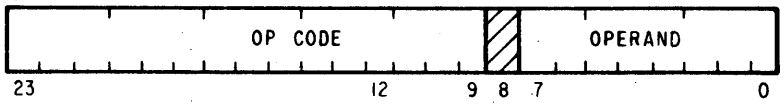
The floating-point number in the D register is added to the number in the X register. The sum replaces the previous contents of the X register.

73.*+X:a	Add Memory to X register	X,Y	AMX



The contents of the Effective Memory Address (EMA) and the next sequential address (EMA+1) are added to the contents of the X register. The sum replaces the previous contents of the X register.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7701.2:o	Add Operand to W register (exponent)	W,Y	AOW

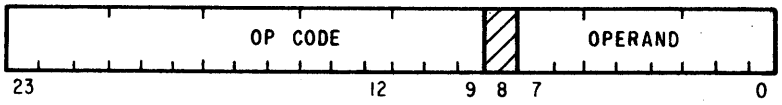


The 8-bit, signed operand is algebraically added to the contents of the W register. The sum replaces the previous contents of the W register.

NOTE

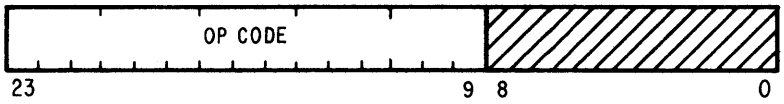
A subtraction may be accomplished by adding a negative operand.

7706.0:o	Add Operand to X register	X,Y	AOX



The signed, 8-bit integer operand is converted to floating-point format and added to the contents of the X register. The sum replaces the previous contents of the X register.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7707.3	Divide A register (integer) into X register	X,Y	DAX

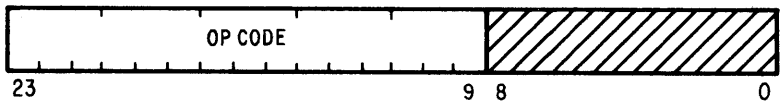


The signed integer in the A register is converted to floating point format. The contents of the X register are divided by the converted number. The quotient replaces the previous contents of the X register.

NOTE

If division by zero occurs, the condition register (Y) is set to overflow, positive and zero, i.e., (Y) = '15.

7710.3	Divide D register (floating-point into X register	X,Y	DDX

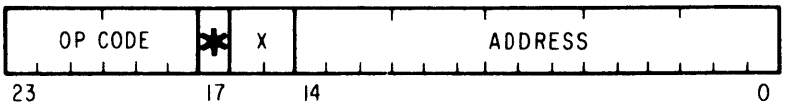


The floating-point contents of the D register are divided into the contents of the X register. The quotient replaces the previous contents of the X register.

NOTE

If division by zero occurs the condition register (Y) is set to overflow, positive and zero, i.e., (Y) = '15.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
76.*+X:a	Divide Memory into X register	X,Y	DMX

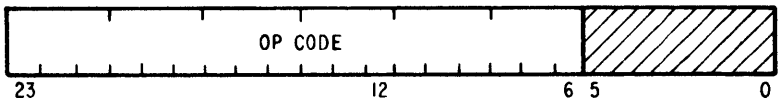


The contents of the X register are divided by the contents of the effective memory address (EMA) and the next sequential address (EMA+1). The quotient replaces the previous contents of the X register.

NOTE

If division by zero occurs, the condition register (Y) will be set to overflow, positive and zero, i.e., (Y) = '15.

7706.3:o	Divide Operand into X register	X,Y	DOX

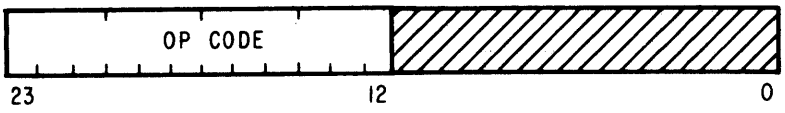


The signed, 8-bit integer operand is converted to floating-point and is divided into the contents of the X register. The quotient replaces the previous contents of the X register.

NOTE

If division by zero occurs, the condition register (Y) will be set to overflow, positive and zero, i.e., (Y) = '15.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7703.	Floating normalize of A register to X register	X,Y	FAX



The signed integer quantity in the A register is converted to a floating-point normalized quantity which replaces the previous quantity in the X register.

NOTES

A positive normalized number will have as the sign and most significant bit the following pattern:

01

A negative normalized number (where the value is not -1) has the configuration

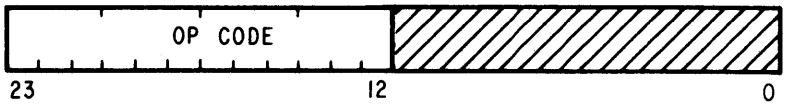
10

A negative normalized number, where the value is -1, results in the mantissa having a bit pattern of all ONEs.

11

If the result is zero, the mantissa will be zero and the exponent will be set to a full scale negative value, i.e., (W) - '201.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7713.	Fix of X register to A register	A,Y	FXA



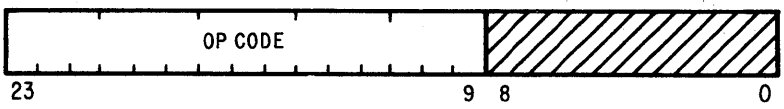
The floating-point number in the X register is converted to a 24-bit signed integer which replaces the previous contents of the A register.

NOTE

If the exponent is greater than 23, the condition register (Y) will be set to overflow, negative and positive, i.e., (Y) = '13.

If the mantissa is negative, a one bit error may result.

7705.0	INverse of X register	X,Y	INX

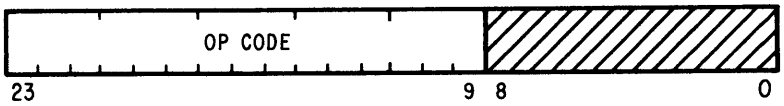


The inverse of the contents $\left[\frac{1}{(X)}\right]$ of the X register replaces the contents of the X register.

NOTE

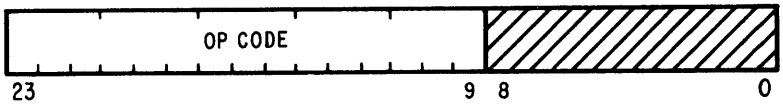
If division by zero occurs, the condition register will be set to overflow, positive and zero, i.e., (Y) = '15.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7707.2	Multiply A register (integer) and X register	X,Y	MAX



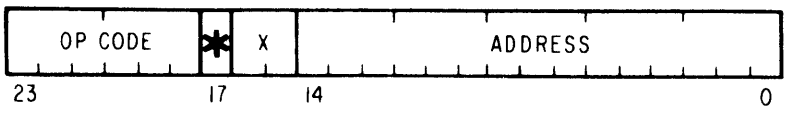
The signed integer in the A register is converted to floating-point format and multiplied by the contents of the X register. The product replaces the previous contents of the X register.

7710.2	Multiply D register (floating point) and X register	X,Y	MDX



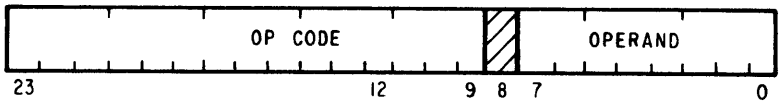
The floating-point contents of the D register are multiplied by the contents of the X register. The product replaces the previous contents of the X register.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
75.*+X:a	Multiply Memory and X register	X,Y	MMX



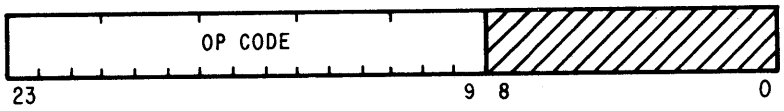
The contents of the X register are multiplied by the contents of the effective memory address (EMA) and the next sequential address (EMA+1). The product replaces the previous contents of the X register.

7706.2:o	Multiply Operand and X register	X,Y	MOX



The signed, 8-bit integer operand is converted to floating point format and is multiplied by the contents of the X register. The floating-point product replaces the previous contents of the X register.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7704.1	Negative of X register to X register	X,Y	NXX

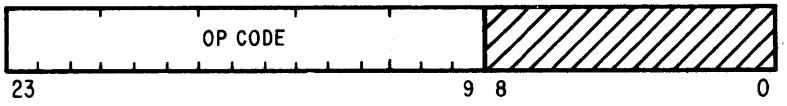


The mantissa in the X register is two's-complemented and the result is loaded into the X register. The Y register is changed to reflect the status of the new quantity.

NOTE

If the bit pattern of the mantissa is 100....0, a one bit error will occur in the result.

7704.0	Positive of X register to X register	X,Y	PXX

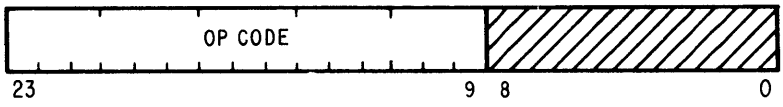


The absolute value of the contents of the X register replaces the previous contents of the X register.

NOTE

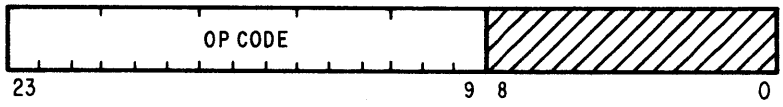
If the bit pattern of the mantissa is 100....0, a one bit error will occur in the result.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7707.1	Subtract A register (integer) from X register	X,Y	SAX



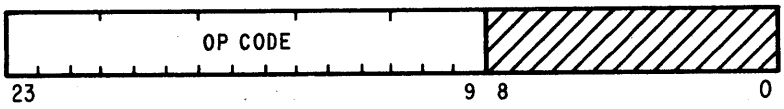
The signed integer in the A register is converted to floating-point format and subtracted from the contents of the X register. The difference replaces the previous contents of the X register.

7710.1	Subtract D register (floating point) from X register	X,Y	SDX



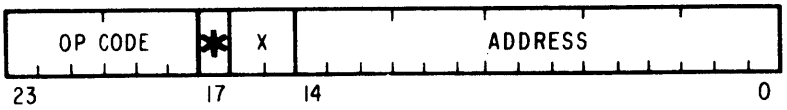
The floating-point contents of register D are subtracted from the X register. The difference replaces the previous contents of the X register.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7705.1	SquarE of X register	X,Y	SEX



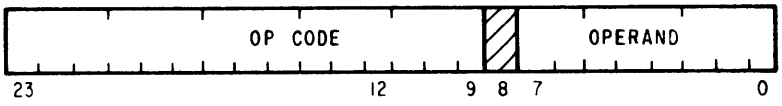
The square of the contents of the X register replaces the previous contents of the X register. (i.e., register X is replaced by X times X.)

74.*+X:a	Subtract Memory from X register	X,Y	SMX



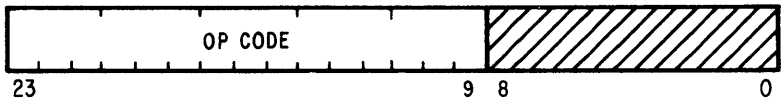
The contents of the effective memory address (EMA) and the next sequential address (EMA+1) are subtracted from the contents of the X register. The difference replaces the contents of the X register.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7706.1:o	Subtract Operand from X register	X,Y	SOX



The signed, 8-bit integer operand is converted to a floating-point format and subtracted from the contents of the X register. The difference replaces the previous contents of the X register.

7705.2	Square Root of X register	X,Y	SRX

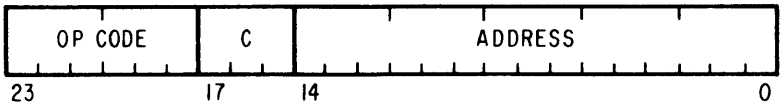


The square root of the contents of the X register replaces the previous contents of the X register.

NOTE

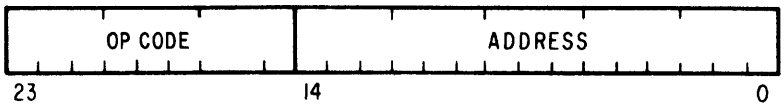
If the contents of register X is negative, the condition register is set to positive, zero, negative and overflow, i.e., (Y) '17.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
63.0:a	Branch on Negative Reset	P	BNR
63.7:a	Branch on Negative Set	P	BNS
64.0:a	Branch on Zero Reset	P	BZR
64.7:a	Branch on Zero Set	P	BZS
65.0:a	Branch on Positive Reset	P	BPR
65.7:a	Branch on Positive Set	P	BPS
77.2:a	Branch on Overflow Reset	P	BOR
77.3:a	Branch on Overflow Set	P	BOS



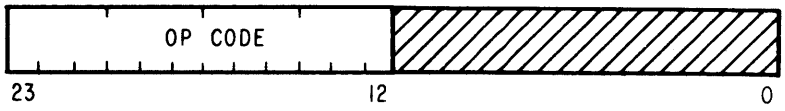
The contents of the Y condition register are tested for the specified condition. If the condition is present, the contents of the P register (current PROGRAM ADDRESS) are replaced by the effective memory address. If the specified condition is not present, the program address advances to the next sequential location (PROGRAM ADDRESS +1).

62.7:a	Branch on SAU Ready	P	BOX



A determination is made as to whether or not the SAU is processing an instruction (the SAU Busy latch is tested). If the SAU is able to process another instruction (i.e., ready) then the contents of the P register (current PROGRAM ADDRESS) are replaced by the effective memory address. If the SAU is currently processing an instruction (i.e., not ready) the program address advances to the next sequential location (PROGRAM ADDRESS +1).

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7712.	Compare D register to X register	Y	CDX

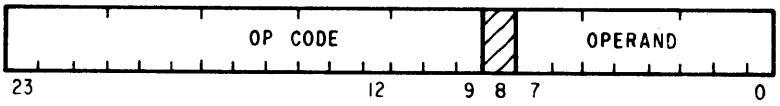


The contents of the D register and the contents of the X register are compared and the Y condition register is set to the status of the result.

Comparison results are as follows:

If X is greater than D;	Y = positive
If X is equal to D;	Y = zero
If X isless than D;	Y = negative

7701.3:o	Compare Operand to W register (exponent)	Y	COW

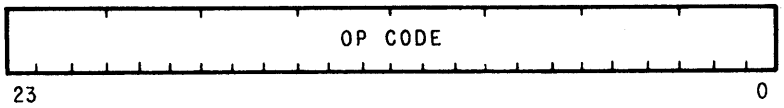


The 8-bit, signed operand and the contents of the W register are algebraically compared and the Y condition register is set to the status of the result.

Comparison results are as follows:

If W is greater than the operand;	Y = positive
If W is equal to the operand;	Y = zero
If W is less than the operand;	Y = negative

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7706.0000	Compare Zero to X register	Y,X	CZX

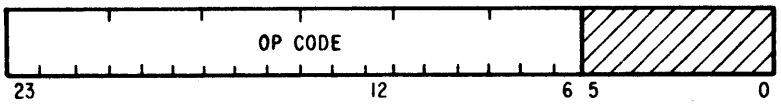


The contents of the X register and floating-point zero are compared and the Y condition register is set to the status of the result.

NOTE

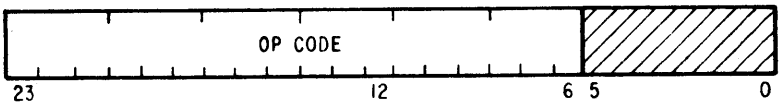
Overflow will result if the mantissa and exponent have the pattern: 110....0/10....0 The least significant bit of X will be set to a 1.

7702.00	Hold SAU overflow Interrupt	None	HSI



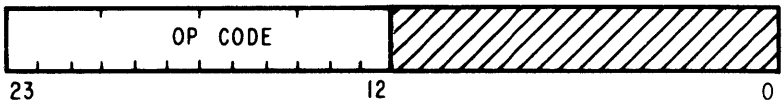
This instruction disarms the overflow/underflow interrupt (Executive trap Group 0, Level 6). The trap remains disarmed until the execution of the release instruction.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7702.01	Release SAU overflow Interrupt	None	RSI



This instruction arms the overflow/underflow interrupt (Executive Trap Group 0, Level 6). When the trap is armed, and not inhibited by an HXI instruction, any SAU operation which causes bit 0 of the Y register to be set (overflow) will generate an interrupt request.

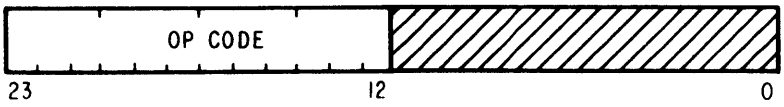
7711.	Interchange D register and X Register	D,X,Y	IDX



The contents of the X register and the D register are interchanged. The Y condition register is set to the status of the X register on completion of the instruction.

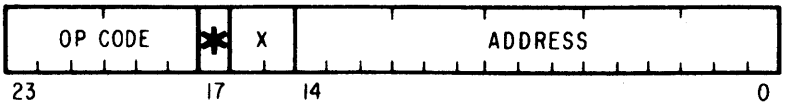
Cycles. . . . . .

7714.	Transfer D register to X register	X,Y	TDX



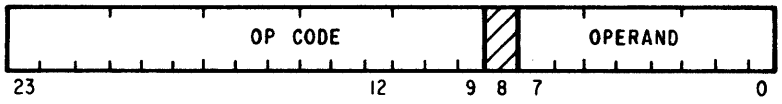
The contents of the D register replace the previous contents of the X register.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
71.*+X:a	Transfer Memory to X register	X,Y	TMX



The contents of the effective memory address (EMA) and the next sequential address (EMA+1) replace the previous contents of the X register. EMA and EMA+1 replace the most signiticant and least significant part of X, respectively.

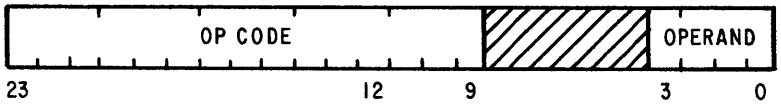
7701.1:o	Transfer Operand to W register (exponent)	W,Y	TOW



The 8-bit, signed operand replaces the previous contents of the W register. All other bits within the X register are unaffected.

The Y condition register is set to the status of the X and XW registers upon completion of the instruction.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7701.0:o	Transfer Operand to Y register	Y	TOY

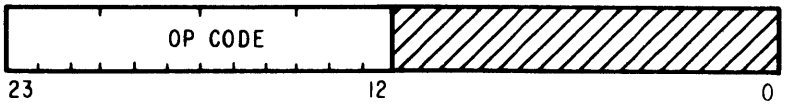


The four bit operand replaces the previous contents of the Y condition register.

Operand definition is as follows:

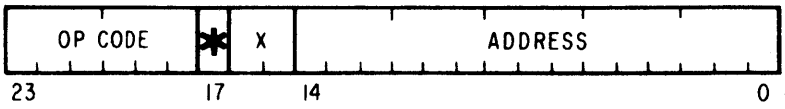
bit 0-ONE = Overflow	ZERO = No overflow
bit 1-ONE = Negative	ZERO = Not negative
bit 2-ONE = Zero	ZERO = not zero
bit 3-ONE = Positive	ZERO = not positive

7715.	Transfer X register to D register	D	TXD



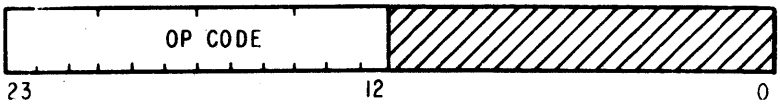
The contents of the X register replaces the previous contents of the D register. The X register is unchanged.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
72.*+X:a	Transfer X register to Memory	M	TXM



The contents of the X register replaces the previous contents of the effective memory address (EMA) and the next sequential address (EMA+1). The most and least significant portions of X are transferred to EMA and EMA+1, respectively.

7700.	Transfer Y register to A register	A	TYA



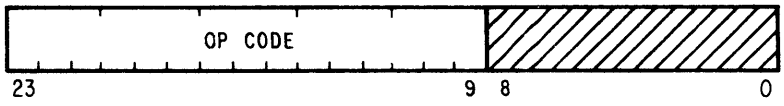
The contents of the Y register are transferred to the A register and the status of the SAU overflow/underflow interrupt will be placed in bit position 6 in the A register.

The following table shows the bit placements of the various Y condition register settings when transferred to the A register.

A Register	Bit Function
Bit 0 = 1	Overflow/Underflow
Bit 1 = 1	Negative
Bit 2 = 1	Zero
Bit 3 = 1	Positive
Bit 6 = 0	SAU Interrupt Enabled
Bit 6 = 1	SAU Interrupt Disabled

All other bits within the A register are set to zero.

INSTRUCTION FORMULA	FUNCTION	REGISTERS AFFECTED	MNEMONIC
7704.2	Transfer Zero to X register	X	TZX



The floating-point representation of zero (0000000000000201) replaces the previous contents of the X register. The Y condition register is unaffected.

APPENDIX A
INSTRUCTION SUMMARY

INSTRUCTION	MNEMONIC	PAGE
Add A register to Memory	AAM	7-12
Add A register to X register	AAX	7-152
Add D register to X register	ADX	7-152
Add E register to Memory	AEM	7-12
Add I register to Memory	AIM	7-12
Add J register to Memory	AJM	7-12
Add K register to Memory	AKM	7-12
Add Memory to A register	AMA	7-10
Add Memory to Byte	AMB	7-11,91
Add Memory to Double register	AMD	7-11
Add Memory to E register	AME	7-10
Add Memory to I register	AMI	7-10
Add Memory to J register	AMJ	7-10
Add Memory to K register	AMK	7-10
Add Memory to X register	AMX	7-152
Add Operand to Byte	AOB	7-13,91
Add Operand to Memory	AOM	7-14
Add Operand to register	• AOR	7-13
Add Operand to W register	AOW	7-153
Add Operand to X register	AOX	7-153
Add Register to Register	• ARR	7-15
Add Unity to Memory	AUM	7-10
Branch when Byte address +1 in I register = 0	BBI	7-35,92
Branch when Byte address +1 in J register = 0	BBJ	7-35,92
Branch indexed by J —— Long	BJL	7-39
Branch and Link I	BLI	7-40
Branch and Link J	BLJ	7-40
Branch and Link K	BLK	7-40
Branch and Link (J) —— Long	BLL	7-40
Branch and Link —— Unrestricted	BLU	7-43,127
Branch on Not Negative	BNN	7-38
Branch on No Overflow	BNO	7-38
Branch on Not Positive	BNP	7-38
Branch on Negative Reset	BNR	7-164
Branch on Negative Set	BNS	7-164
Branch on Not Zero	BNZ	7-38
Branch On Negative	BON	7-38
Branch On Overflow	BOO	7-38
Branch On Positive	BOP	7-38
Branch on Overflow Reset	BOR	7-164
Branch on Overflow Set	BOS	7-164
Branch on SAU Ready	BOX	7-164
Branch on Zero	BOZ	7-38
Branch on Positive Reset	BPR	7-164
Branch on Positive Set	BPS	7-164
Branch on Reset interrupt —— Long	BRL	7-42,31

INSTRUCTION	MNEMONIC	PAGE
Query Bits of Byte	QBB	7-100,144
Query Bit of H	QBH	7-119
Query Bit of Memory	QBM	7-123
Query Sense Switches	QSS	7-145
Right shift Arithmetic A	RAA	7-68
Right shift Arithmetic Double	RAD	7-68
Replace Byte in Memory	RBM	7-78,99
Release Clock Time	RCT	7-147
Right shift Logical A	RLA	7-69
Right shift Logical Double	RLD	7-69
Release Processor Time	RPT	7-146
Right Rotate A	RRA	7-70
Right Rotate Double	RRD	7-70
Round of Register to Register	• RRR	7-28
Release SAU overflow Interrupt	RSI	7-167
Release eXternal Interrupts	RXI	7-133
Subtract A register from X register	SAX	7-161
Subtract D register from X register	SDX	7-161
SquarE X register	SEX	7-162
Subtract Memory from A register	SMA	7-29
Subtract Memory from Byte	SMB	7-30
Subtract Memory from Double	SMD	7-30
Subtract Memory from E register	SME	7-29
Subtract Memory from I register	SMI	7-29
Subtract Memory from J register	SMJ	7-29
Subtract Memory from K register	SMK	7-29
Subtract Memory from X register	SMX	7-162
Subtract Operand from Byte	SOB	7-31,101
Subtract Operand from Register	• SOR	7-31
Subtract Operand from X register	SOX	7-163
Square Root —— Extended	SRE	7-33
Subtract Register from Register	• SRR	7-32
Square Root	SRT	7-33
Square Root of X register	SRX	7-163
Transfer A register to Memory	TAM	7-88
Transfer Byte to Memory	TBM	7-86,101
Transfer Double to Limit registers	TDL	7-128
Transfer Double to Memory	TDM	7-86
Transfer D register to X register	TDX	7-167
Transfer Double to group 1	TD1	7-133
Transfer Double to group 1	TD4	7-135

● Mnemonic represents a family of instructions