CONTROL DATA
CORPORATION

CONTROL DATA®
3500 COMPUTER SYSTEM

REFERENCE MANUAL

3500 CHARACTERISTICS

- Stored-program, solid-state, scientific and business data processing computer
- Integrated circuitry, 50-PAK logic module construction
- Transistor Current Switch (TCS) logic
- Upwards compatability with the 3100, 3200, and 3300 computer systems
- Parallel mode of operation
- Character and word addressing (4 characters per word)
- Address modification (indexing) and indirect addressing
- Parity chacking on data and addresses
- 28-bit storage word (24 data bits and 4 parity bits)
- Nonvolatile magnetic core storage
- Complete cycle time: 0.900 microseconds
- Storage access time: 0.500 microseconds
- Storage sharing and selected storage protection
- Single-precision (24-bit) and double precision (48-bit) binary arithmetic
- Floating point arithmetic
- Logical and sensing operations
- Block operations (I/O oriented buffer priority)
- Trapped instruction processign
- Complete interrupt system
- 64-word register file
- Real-time clock (1.0 millisecond incrementation)
- 12-bit and 24-bit bidirectional data channels
- Sit-down operator's console featuring on-line typewriter and complete display and control system
- Maintenance test facilities for arithmetic and I/O channels
- Compatible I/O media include magnetic tape, disk file, punched cards, paper tape, and printed forms
- Options include:

  Business oriented Moves, Searches, Edit, Compares, Conversions, and BCD arithmetic

  Address relocation hardware

  Real-Time Interrupt Processor (RIP)

  Memory expansion to 262, 144 words (over 1 million characters)

  Additional 12-bit and 24-bit data channels

  Complete Selection of advanced peripheral equipment

  Register/Page File Parity

CONTROL DATA
CORPORATION

CONTROL DATA®
3500 COMPUTER SYSTEM

REFERENCE MANUAL

RECORD of REVISIONS

REVISION	NOTES
01	Training edition. Company distribution only.
(12-15-66)	
02	Preliminary edition. This printing obsoletes previous editions.
(7-14-67)	
A	Final Edition. This printing obsolete all previous editions.
(11-15-69)	
B	Manual revised; includes Engineering Change Order 33777, publications change only. Cover, Inside
(5-14-73)	front cover, v, 1-4, 1-5, 1-6, 1-7, 1-8, 1-12, 1-13, 2-6, 4-1, 4-8, 4-9, 5-2, 5-8. 6-51, 6-83 and
	7-2 are revised. Page 3-8 is added.
C	Manual revised; includes Engineering Change Order 34320. Pages 6-65, 6-69, and 7-9 are revised.
(2-4-74)	
D	Manual revised; includes ECO 35032. Pages 5-1, 5-6, 6-69, 6-70, 6-136, 6-138, 6-155, and
(8-7-74)	6-158 revised.

FORM CA230 REV. 1-67

Pub No. 60200300

Printed in United States of America

Address comments concerning this manual to:

Control Data Corporation
Technical Publications Department
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual.

PREFACE

This manual provides information for the machine language use of the CONTROL DATA® 3500 Computer System. The purpose of this manual is to describe the capabilities of the hardware and to note programming restrictions.

COMPASS mnemonics are used to abbreviate titles of instructions; however, no software systems are used in describing instructions. Detailed descriptions for those systems in operation are available in the appropriate software reference manuals.

Programming information for most available peripheral equipment is contained in the 3000 Series Peripheral Equipment Reference Manual. For latest publications and revisions see the Control Data Literature Catalog.

CONTENTS

FIGURES

TABLES

3500 COMPUTER SYSTEM

1. GENERAL SYSTEM DESCRIPTION

INTRODUCTION

The CONTROL DATA® 3500 Computer System is a general-purpose digital computing system of advanced design which features integrated circuitry and provides both scientific and business users with extremely high performance. A fully expanded system includes double precision and floating point arithmetic capabilities. It also has a comprehensive business data processing instruction set, an address relocation system that permits multiprogramming and full utilization of an expanded storage system, and a Real-Time Interrupt Processor which provides up to 64 additional priority interrupt lines and decreases interrupt response time.

All programs written for CONTROL DATA® 3100, 3200, and 3300 Computer Systems can be processed by a 3500. The I/O characteristics for the 3500 are identical to CONTROL DATA® 3000 Series computers. A wide selection of proven peripheral equipment is available for use in a 3500 system, including many new and advanced equipments.

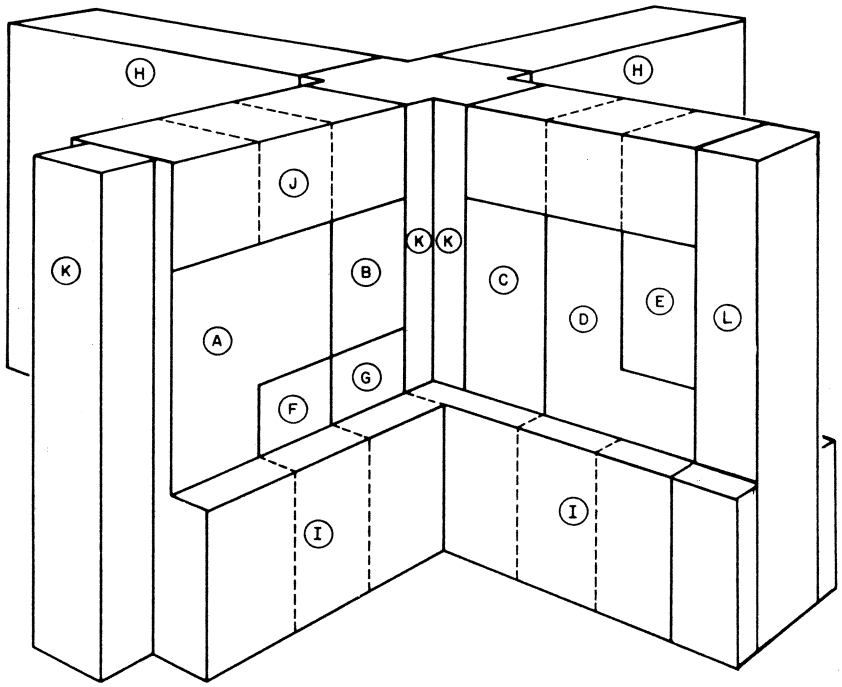
Efficient software operating systems are available to complement the hardware. The Multiple Access Shared Time Executive Routine (MASTER) is a mass-storage-oriented multiprogramming executive operating system that can handle a multitude of real-time applications. It features multi-access capabilities for multiple on-line stations, for remote batch processing and for file manipulation. MASTER's greatest feature is that it increases total system throughput by overlapping activities of many tasks on compute modules and data channels. MASTER considers multiple jobs simultaneously so that activities can be found for processors whenever they become available.

The Mass Storage Operating System (MSOS) provides serial batch processing with concurrent execution of a special-purpose interrupt-controlled program. The operating system, software library, and user data areas are allocated to mass storage. MSOS features real-time priority-interrupt handling and a comprehensive mass storage input/output system which is available to user programs. Certain non-standard real-time devices can be added to both operating systems; thus,the user can tailor the system to fit his needs. Detailed descriptions of these and other software systems are available upon request from the nearest Control Data sales office.

This manual describes the 3500 in terms of the Central Processor and its storage, I/O, and interrupt systems, and provides the programmer with a complete description of the hardware instruction set. Reference information is available in the Appendixes.

3500 COMPUTER CONFIGURATION

The circuitry used in the 3500 allows a convenient packaging of logic components and assemblies. Individual integrated circuits are mounted on multi-layer printed circuit boards, forming 50 PAK logic modules. Fifty PAK's related logic functions form assemblies called logical units. The two wings that form the main computer contain the logical units for the Central Processor and I/O Channels, and also house the necessary power supply and cooling equipment. Figure 1-1 illustrates a typical 3500 configuration.



Ⓐ BLOCK CONTROL (INCLUDING I/O CHANNELS)
Ⓑ RELOCATION
Ⓒ MAIN CONTROL
Ⓓ ARITHMETIC
Ⓔ BUSINESS DATA PROCESSOR (BDP)
Ⓕ COMPUTER MAINTENANCE PANEL
Ⓖ CONSOLE DISPLAY INTERFACE
Ⓗ STORAGE MODULES
Ⓘ BLOWERS AND POWER SUPPLIES
Ⓙ POWER CONTROL PANELS (ONE PER COLUMN)
Ⓚ CONNECTOR PANELS
Ⓛ REAL-TIME INTERRUPT PROCESSOR

Figure 1-1. 3500 Configuration

Central Processors

Several central processor models are available for 3500 systems. All central processors are basically similar but some models have additional features designed for special applications.

Some of the most important central processor functions are:

- Performs all 24-/48-bit precision fixed point arithmetic
- Performs all floating point arithmetic
- Executes business data processing instruction set (some models)
- Executes Boolean (logical) and decision-making instructions
- Enables 24-bit word and 6-bit character data manipulation
- Controls real-time clock and register file references
- Controls standard search/move operations, external equipment and typewriter I/O
- Performs address relocation (some models)
- Recognizes and processes interrupts
- Provides memory protection
- Processes time-critical interrupts on a real-time basis (some models)

3504-1 Central Processor

This model performs all functions listed above except for real-time interrupt processing. It contains five functional sections:

- Main Control Section
- Arithmetic Section
- Block Control Section
- Address Relocation Section
- Business Data Processor (BDP)

3514-4 Central Processor

This model is similar to the 3504-1 with two major differences:

1. An optional 3522 Real Processor can be added. This unit adds up to 64 additional external interrupt lines and significantly reduces the time required to process interrupts from external sources.
2. The business data processing instruction set in the 3514-4 is somewhat different from the 3504-1 BDP instructions.

3514-3 Central Processor

This model is identical to the 3514-4 except that the Business Data Processor is not present. It includes the Main Control, Arithmetic, Block Control and Address Relocation sections.

3514-2 Central Processor

This model is identical to the 3514-4 except that the Address Relocation section is not present. It includes the Main Control, Arithmetic, Block Control and Business Data Processor sections.

3514-1 Central Processor

This model is identical to the 3514-4 except that neither the Business Data Processor nor Relocation section is installed. It includes the Main Control, Arithmetic and Block Control sections.

The optional 3522 Real Time Interrupt Processor may be added to any of the 3514 Central Processors but not to the 3504-1. Standard Option 10288 Register/Page File Parity Checking, may be added to the 3504 or 3514 Central Processors.

Magnetic Core Storage

The basic magnetic core storage* for the 3500 system is the CONTROL DATA® 3502-32 Storage Module. The system can be expanded from this minimum of 32,768 words to a maximum of 262,144 words (1,048,576 characters). The number following the dash corresponds to the size of the storage module:

3502-32 Storage Module ---------- 32,768 words
3502-65 Storage Module ---------- 65,536 words

3502-262 Storage Module ---------- 262,144 words

The storage shown in the configuration (Figure 1-1) consists of 65K words. Other storage configurations are available, including those with modules as stand-alone units. An option is available to expand the number of storage access channels to a maximum of four per module; one channel per module is standard. Refer to Section 2 for detailed storage information.

---

* The word storage is used synonymously with memory throughout the text, and both refer to magnetic core storage unless otherwise stated.

Input/Output Channel

The input/output channel available in the 3500 system is the CONTROL DATA® 3507-1 Communication Channel. The channel operates in either of two modes to fit the particular application. A maximum of eight channels may be incorporated into a single-processor system.

In the 12-bit mode, the channel provides a 12-bit parallel data path with 12- to 24/24- to 12-bit assembly/disassembly capabilities. In 24-bit mode the channel provides a 24-bit parallel data path for direct high-speed word transfers.

The channel conforms to the standard I/O specification for CONTROL DATA 3000 Series Computers. Refer to Section 3 for the interface signals used and for detailed information on the channel's two operating modes.

Real-Time Interrupt Processor (RIP)

The optional RIP available for 3514 central processors is the CONTROL DATA 3522 Real-Time Interrupt Processor. The RIP provides the system with up to 64 additional priority interrupt lines. Real-time interrupt capability provided by the RIP includes the following:

1. 16, 32, 48, or 64 real-time interrupt inputs to the RIP
2. Fixed interrupt priority 1 through 64
3. Automatic store of central processor operational registers when interrupting on lines 00 - 17_8
4. Automatic lockout of current and lower priority interrupts upon interrupt recognition
5. Unique interrupt address for each interrupt line.

The addition of an optional RIP enhances the system for real-time applications by providing reduced interrupt response time. Refer to Section 4 for detailed information regarding the Real-Time Interrupt Processor.

REGISTER/PAGE FILE PARITY CHECKING OPTION

Standard Option 10288 provides error detection in the register and page file and also provides detection of multiple parity errors and parity errors which occur when abnormal interrupt sensing is disabled. A corresponding coded interrupt is generated upon cover detection, adding three interrupts to abnormal interrupt system.

INTERNAL ORGANIZATION

Computer Word Format

The standard 3500 computer word (Figure 1-2) consists of 24 bits (binary digits) grouped into four 6-bit characters. Parity is generated for each data character prior to entering storage; thus, the word length in storage is lengthened to 28 bits to accommodate the parity bits.

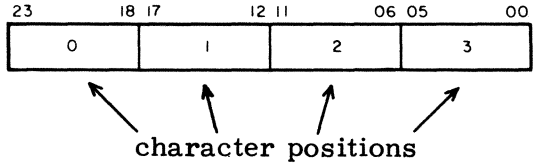


Figure 1-2. Computer Word Format

Central Processor

The Central Processor contains the necessary logic to control all sequences and synchronize internal operations within the storage modules and data channels. The processor is logically organized into five sections: Main Control, Block Control, Arithmetic, Business Data Processing (BDP), and Relocation. The BDP and the address relocation feature are optional in the 3514 but they are standard in the 3504.

Main Control Section

Main Control initiates execution of all instructions and enables subsequent action as required in either the Arithmetic, BDP, or Block Control sections for instruction completion.

The section has access to core storage via the Relocation section, and interfaces with the operator's console to provide synchronization and control of manual switches and keyboard operations. The principal registers in the section are described as follows:

P Register (15 bits): The P register is the Program Address Counter. It provides program continuity by generating in sequence the basic storage addresses for retrieving the individual instructions. Normally P is incremented by 1 at the completion of each instruction. The address, which may be modified via the Relocation section, is sent to core storage to select the next instruction. On some models, blue background lights indicate parity errors.

A skip exit advances the count in P by 2, bypassing the next sequential instruction and executing the following one. For a jump exit, the execution address portion of the jump instruction is entered into P and is used to specify the starting address of a new sequence of instructions.

F Register (24 bits): The F register receives the instruction from storage and retains it during interpretation and execution. During execution, the address portion of the instruction may be modified as follows:

- Indexing - A quantity in one of the index registers (B^b) is added to the lower 15 bits of F for word-addressed instructions, or to the lower 17 bits of F for character-addressed instructions.
- Indirect Addressing - The lower 18 bits of F are replaced by the lower 18 bits of the location specified by the original address. For load and store index instructions, bits 00-14 and bit 17 of F are replaced by the respective bits from the location specified by the original address.

The F register shares a display on the console with the Communication register. The F register is displayed whenever the keyboard is inactive and the computer is not in the Go mode.

C Register (24 bits): Quantities to be entered into the A, Q, B^b, or P registers, the Page File, Register File, or storage from the entry keyboard are temporarily held in the Communication (C) register until the TRANSFER switch is pressed. If an error is made while entering data into the Communication register, the KEYB CLR switch may be used to clear this register.

The C register displays words read from the Page File, Register File, or storage during certain manual operations. The (C)* are displayed on the console whenever the keyboard is active.

* The parentheses are an accepted method for expressing the words "the contents of" (in this case, "the contents of the C register").

B^b Registers (15 bits): The three index registers, B^1, B^2, and B^3, are generally used for counting and indexing operations. When indexing is specified in an instruction, the contents of one of these registers is added to the original word or character address, thus forming a new execution address. The registers can be used for testing jump and skip conditions during a program, and for inter-register transfer operations.

Upon completion of certain business-oriented instructions, the B^3 register is used to record the C field character count, BCD fault information, and other operating conditions (see Section 6). The index registers and P register share a display area on the console.

Instruction State Register (3 bits): The ISR is used during Executive Mode operations to select one of eight areas within the relocation memory (Page Index File) that is addressed during an instruction memory reference. The (ISR) are appended to the basic 15-bit program address (P) to form an original 18-bit address. The register can be altered only during Monitor state of Executive mode. Refer to Section 5 for specific information on the use of this register.

Operand State Register (3 bits): The Operand State register (OSR) is used like the ISR except during operand references. The register can be altered only during Monitor state of Executive mode. Refer to Section 5 for specific information on the use of this register. Blue background light is on when OSR is active.

Channel Index Register (3 bits): During Monitor state of Executive mode, the contents of the Channel Index register (CIR) are logically ORed (inclusive OR function) with the channel designator (ch) bits for the following instructions:

73-76	I/O instructions
77.0	Connect
77.1	Select Function
77.2	Sense External Status
77.2	Copy External Status
77.3	Sense Internal Status
77.3	Copy Internal Status
77.4	Sense Interrupt

This permits instructions to be written for Channel 0 and allows the monitor program to assign an available channel by altering the (CIR).

The (CIR) can be transferred by instruction to the lower 3 bits of the A register and vice versa. A switch is provided on the console for displaying (CIR) in the lowest digit position of the P register display.

Condition Register (6 bits): Bits in the Condition register (CR) are used as flags to initiate computer action and to record current operating conditions during Executive mode. With the exception of bit 04, the register is not used during non-executive mode operations.

All register bits can be set or cleared with the ACR (77.634) instruction; selected bits are set or cleared by individual instructions and conditions. Refer to Section 4 for special considerations involving the register during interrupt processing. The register bit assignments are listed below:

Bit 00	Boundary Jump
Bit 01	Destructive Load A
Bit 02	Operands Relocated Using OSR
Bit 03	Program State Jump
Bit 04	Interrupt System Enabled
Bit 05	Program State

Block Control

Block Control is an auxiliary control section that is chiefly responsible for the following operations:

- I/O data transfers (including typewriter I/O)
- Search/Move
- Interrupt and status sensing
- Temporary storage (Register File)
- Real-time clock updating

Block Control shares access to storage with Main Control (via the Relocation section) and is capable of operating in parallel with the Arithmetic, BDP, and Main Control sections. Block Control interfaces directly with the console typewriter and the data channels.

The principal register within Block Control is the 24-bit D_4 register. All data passing through Block Control during I/O operations is temporarily held in D_4 Data is shifted in conjunction with the register to enable character and byte manipulation.

Register File: The Register File is a rapid-access flip-flop matrix containing 64 28-bit registers. It is used by Block Control for temporary storage and for address manipulation and control during block operations. Although the programmer has access to each register (lower 24 bits only) in the file with the inter-register transfer (53) instruction, reference should be made to Executive Interrupt in Section 4 for restrictions on the use of the file. Register file assignments are listed in Table 1-1.

Manual read/write operations with the Register File are possible using the Breakpoint switches on the operator's console (see Section 7).

A parity bit is generated and written for each transfer (27 bits) into the file. Parity is checked any time the file is referenced, if the PARITY STOP or PARITY INTERRUPT switch is active.

TABLE 1-1. REGISTER FILE ASSIGNMENTS

REGISTER NUMBERS	REGISTER FUNCTIONS
00-07	Modified I/O instruction word (P+1) containing the current character address (Channel 0-7 control)
10-17	Modified I/O instruction word (P) containing the last address ±1, depending on the instruction (Channel 0-7 control)
20	Search instruction word containing the current character address (search control)
21	Move instruction word containing the current source character address (move control)
22	Real-time clock, current time (binary count in milliseconds)
23	Current character address (typewriter control)*
24-27	Temporary storage
30	Instruction word containing the last character address + 1 (search control)
31	Instruction word containing the current destination character address (move control)
32	Real-time clock, interrupt mask
33	Last character address + 1 (typewriter control)*
34-77	Temporary storage

*Reference CTI and CTO instructions for format requirements of registers 23 and 33.

Block Control Priority: Priority to Block Control is I/O oriented, with Channel 0 having highest priority. Any lower numbered channel has priority over a higher numbered channel and therefore should be used to service the fast I/O devices in a system. Once a channel request is honored, the request scanning process is reset to Channel 0. Requests for I/O are honored until a maximum number of active operations limit further honoring of requests, at which time Block Control is operating at its maximum rate.

The main program requires access to Block Control to initiate I/O and S/M buffer operations. This initiation consists of storing modified forms of the instruction words in the Register File, hence providing available operating information (addresses, channel number, etc.) to Block Control. Once the desired I/O operations have been initiated, high priority is not given to further initiation. Table 1-2 lists the different buffer groups, in their normal order of priority, and also shows the priority within groups.

TABLE 1-2. BUFFER GROUPS

GROUP 1	GROUP 2	GROUP 3
Channel 0	Channel 4	Program
Channel 1	Channel 5	Real-time Clock
Channel 2	Channel 6	Typewriter
Channel 3	Channel 7	Search/Move

If the I/O system has not reached maximum transfer rate, the requests in Group 3 will be honored in their normal order of priority. However, since Block Control provides for maximum bursts of I/O without interruption, the honoring of requests for updating the real-time clock, clearing the channels via instruction (CLCA and INCL), and servicing typewriter I/O is delayed as long as possible so that burst rates may be achieved. These requests are then serviced with the highest priority after the delay. The flow chart in Figure 1-3 illustrates the priority scheme, including the special delayed requests. Note that Master Clear, Internal Clear, and External Clear requests are honored before any buffer requests.

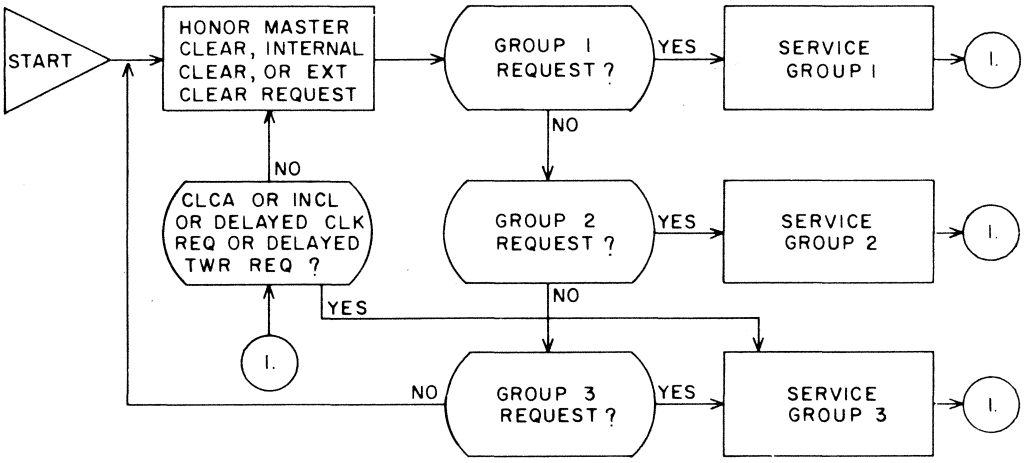


Figure 1-3. Block Control Priority Scheme

Real-Time Clock: The real-time clock is a 24-bit counter that is incremented each millisecond to a maximum period of 16,777,216* milliseconds. After reaching its maximum count the clock returns to zero and the cycle is repeated. The clock, controlled by 1 kilocycle signals, starts as soon as power is applied to the computer. The current time is stored in register 22 of the Register File and is removed for updating and comparison with register 32 once every millisecond. When necessary the clock may be reset to any 24-bit quantity by loading A and then transferring (A) into register 22.

A Real-Time Clock Interrupt condition is generated when the current time (register 22) equals the time specified by the clock mask (register 32). Performing a master clear

*16,777,216 milliseconds equal approximately 4 hours and 40 minutes.

does not affect the clock count. For a special case involving the real-time clock, refer to the Priority Pause (PRP) instruction in Section 6.

Arithmetic

The Arithmetic section works in conjunction with the BDP and Main Control in performing all arithmetic (except address and index operations) and logical and shifting operations. This section contains the main operational registers. Those discussed here are displayed on the console. A and Q can be altered from the keyboard.

A Register (24 bits): The A register (accumulator) is the principal arithmetic register. Some of the more important functions of this register are:

- Most arithmetic and logical operations use the A register in formulating a result. The A register is the only register with provisions for directly adding its contents to the contents of a storage location or another register.
- The A register may be shifted to the right or left separately or in conjunction with the Q register. Right shifting is end-off; the lowest bits are discarded and the sign is extended. Left shifting is end-around; the bit leaving the highest-order position appears in the lowest-order position after each shift; all other bits move one place to the left.
- I/O operations with the register are available (word and character).

Q Register (24 bits): The Q register is an auxiliary accumulator register and is generally used in conjunction with the A register. The principal functions of Q are:

- Providing temporary storage for the contents of A while A is used for another arithmetic operation.
- Forming a double-length register, AQ, for use in arithmetic operations.
- Shifting to the right or left, separately or in conjunction with A.
- Serving as a mask register.

E Register (48 bits): The E register is used with the A and Q registers during floating point and double precision arithmetic and during transfer operations. The register is divided into two equal parts, E_U and E_L, each composed of 24 bits. It is used during arithmetic as follows:

- 48-bit precision multiplication; holds the lower 48 bits of a 96-bit product.
- 48-bit precision division; initially holds the lower 48 bits of the dividend; upon completion, holds the remainder.
- Floating point multiplication; holds the residue of the coefficient of the 48-bit product.
- Floating point division; holds the remainder.
- Floating point addition and subtraction; holds the residue.

Business Data Processing Section

Main Control initiates the optional BDP section for execution of a unique set of business-oriented instructions. With this instruction set, variable field length data processing is added to the capabilities of the Central Processor. All of these instructions require three computer words and are stored consecutively, at the current address defined by P, P+1, and P+2. Upon exit from one of these instructions, the BDP relinquishes program control to Main Control for continued program execution.

Descriptions of the BDP instructions and detailed programming information are provided in Section 6.

Relocation Section

All storage access for the Central Processor is through the Relocation section. Priority is established between Block Control and Main Control, and address relocation is performed if required. Constants are held in an optional rapid-access flip-flop matrix called the Page Index File, which is referenced during relocation to obtain physical page address information. Refer to Section 5 for a detailed discussion of the relocation process. The Relocation section also generates and checks parity on all data and address transmissions with storage. A parity bit is generated and written for each word written into the page file (12 bits). Parity is checked any time the file is referenced, if the PARITY STOP or PARITY INTERRUPT switch is active. Storage parity checking is described in Section 2.

Operational Modes

The 3500 Central Processor operates in either the Non-Executive or Executive mode. A definite instruction set and storage capacity is associated with each operating mode. The BDP instructions can be executed in either mode, provided BDP MODE is selected on the console.

Non-Executive Mode

In Non-Executive mode, the 3500 will execute the 3100/3200 instruction set, provided BDP MODE is not selected. If BDP MODE is selected the optional 3500 BDP instructions are executed; however, termination of these instructions due to an Illegal Write condition is not possible.

Maximum storage capacity in Non-Executive mode is 32, 768 words and relocation of addresses does not occur. The upper 3 bits of the 18-bit storage address are forced to zero. During Non-Executive mode, instructions exclusive to the 3500 become No-Operation (No-Op) instructions. These No-Ops are listed in Section 6.

Executive Mode

Monitor State: The initial operating state of Executive mode is the Monitor state. Monitor state is selected by pressing the EXECUTIVE MODE switch on the operator's console, or by master clearing the processor while stopped in Executive mode.

All 3500 instructions can be executed during Monitor state. The Monitor state enables full use of the address relocation and I/O systems, and becomes the normal operating state for processing all interrupts occurring in Executive mode.

Program State: In Program state, the Central Processor operates at its highest efficiency by restricting itself to actual computations, excluding I/O and certain other instructions. The Program state condition is recorded by setting bit 05 of the Condition register. The following instructions cannot be executed during Program state and cause an Executive interrupt if attempted:

- Halt (00.0)
- 71-77 (except 77.72 and 77.71)
- Any inter-register transfer into registers $00\text{-}37_8$ of the Register File.

As the Executive interrupt is recognized, the computer reverts to Monitor state, at which time the interrupt-causing instruction can be executed.

BDP Mode

Business data processing instructions (64-70) can be executed at any time (in either Monitor or Program state of Executive mode or Non-Executive mode) provided the BDP option is present and the BDP mode has been selected via the BDP MODE console switch. The various moves, searches, code conversions, and BCD arithmetic instructions are executed by the Central Processor. Refer to Business Data Processing Instructions in Section 6 for detailed programming information.

With the BDP mode not selected, all instructions with octal codes 64-70 are trapped. Any program written for the CDC® 3100/3200 Computer Systems (including those with BCD instructions) can then be executed via software simulation, provided the proper operating system is used.

Status and Interrupts

The programmer can monitor status conditions during program execution by using sense instructions. External and internal status conditions are selected by the appropriate sensing mask; program execution continues at location P+1 if a selected condition is active, or skips to location P+2 if inactive.

A programmer can set his program to automatically enter an interrupt routine by selecting the desired interrupt condition(s) with an interrupt mask. If the interrupt system is enabled, the program will interrupt whenever a selected condition is recognized. The current program address (P) and a code representing the interrupt condition are stored at a specified address, and control is given to an interrupt routine.

Other interrupts occur automatically to provide protection and are not under control of the programmer. Refer to Section 4 for detailed interrupt and status information.

Parity Checking System

The Central Processor provides parity checking on address and data transmissions to and from core storage (refer to Section 2) and on data transmissions to and from the optional RIP. This checking is always performed, regardless of the control section that is accessing storage. During I/O operations, the processor, in conjunction with the peripheral equipment, checks parity on all external data transfers. I/O parity is discussed in detail in Section 3.

Odd parity is used during all parity checking; that is, the total number of "1" bits in the quantity checked plus the parity bit must be odd. In some models parity checking is provided for all register and page file references.

PERIPHERAL EQUIPMENT

A wide variety of peripheral equipment is available for use with the 3500 Computer. All peripheral equipment available for CONTROL DATA® 3000 Series Computer Systems* can be attached to the 3500 data channel. For programming instructions, as well as a list of function and status response codes, refer to the 3000 Series Peripheral Equipment Reference Manual, Pub. No. 60108800.

* 3100, 3200, 3300, 3400, 3500, 3600, and 3800 systems

2. STORAGE SYSTEM

GENERAL INFORMATION

The Magnetic Core Storage (MCS) system provides high-speed random access core storage for the 3500 Central Processor or other access devices. A maximum storage capacity of 262,144 words may be incorporated into a single-processor system. The standard storage word is 28 bits in length; 24 data bits and 4 parity bits. The data bits (bits 00-23) may contain a single 24-bit instruction word, part of a two-or three-word instruction, a 24-bit operand, or part of a larger operand. Bits 24-27 are the parity bits associated with each of the four characters in the computer word. The parity checking system is described later in this section.

3502 STORAGE MODULE

The 3502 Storage Module has a cycle time of 900 nanoseconds and an access time of 600 nanoseconds. The minimum storage configuration is the 32,768-word 3502-32 Storage Module. If the system is operating in Non-Executive mode, additional storage is not addressable. In Executive mode, storage modules can be added to reach a maximum of 262,144 words. Additional 3502 modules can be added to increase storage in 32K increments.

There are four 8K storage banks within each 32K module. The banks are organized into independent 16K groups (upper and lower) to facilitate addressing and optimize operations. A maintenance panel on each storage module provides the controls and meters for the module's logic voltage and its cooling system.

ACCESS CHANNELS

Each 32K module of storage is equipped with one standard access channel for servicing storage requests. The channel provides independent access to each 16K group within the 32K module. Up to three additional channels may be added per 32K. For example, with the maximum of four channels present, two 3500 Central Processors and two additional devices may access a single 32K of storage.

Access to each 16K group is controlled by a channel scanner. The scanner monitors the access channels present and stops at the channel with an active request to allow a one-word reference. The scanner releases and advances to the next channel before the cycle is completed.

The control logic for the 8K banks within a 16K group allows overlapping references from one bank to the other. It is therefore possible to have two concurrent references overlapped with the two other references within the same 32K module.

STORAGE ADDRESSING

Each 16K group is identified by four binary switches which correspond to bits 14 through 17 of the storage address. A 16K group responds only if the switches match the corresponding address bits. Since each channel may have access to either 16K group within a 32K module, there are two sets of switches per channel. Thus, the groups can be designated differently for each of the four possible channels.

During Non-Executive mode operations, the upper 3 bits of address are forced to remain zero, limiting access to two 16K groups. In Executive mode, the contents of the 3-bit Instruction State Register (ISR) or Operand State Register (OSR) are appended to the basic 15-bit word address. The resulting 18-bit address provides for addressing up to 262,144 words.

Figure 2-1 illustrates the addressing scheme used for the 3502 storage system. With this scheme even and odd addresses are actually located in different 8K banks.

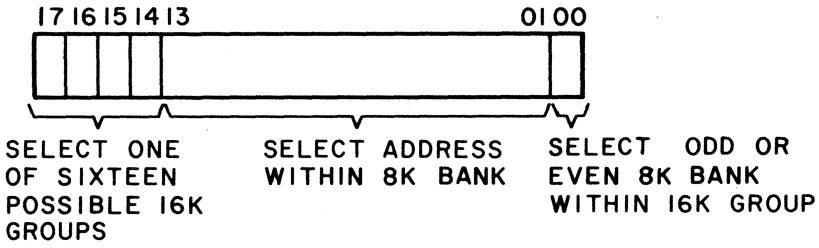


Figure 2-1. 3502 Addressing Scheme

PARITY CHECKING

The Central Processor and storage check parity on all storage addresses and data transmitted to and from storage; odd parity is used. Unless the PARITY STOP or PARITY INTERRUPT switch on the console is active, all parity error indications are ignored. These parity condition switches are described in Section 7.

Address Parity

Logic within the Relocation section generates a parity bit to accompany the 18-bit address sent to storage. The selected storage module generates new parity for the address and compares it with the transmitted parity. If a parity error is detected, a Parity Error signal is returned to the Central Processor. If a Write into storage operation is specified when the address parity error occurs, the Write signal is changed to a Read in the selected storage module and the Write operation does not occur.

Data Parity

During a write into storage operation, the Relocation section generates and sends to storage a parity bit for each 6-bit character. The parity bit(s) is placed in storage with the data. Storage generates parity only during write address (Address mode) operations, and then only for Character 1.

When storage is referenced during a read from storage operation, all parity bits are returned to the Relocation section with the data word. At that time the data is checked for correct parity, and a Parity Error signal is generated if a parity error is detected.

STORAGE PROTECTION

It is often desirable to protect the contents of certain storage addressing against alteration during the execution of a program. There are three types of storage address protection: 1) permanent protection, 2) selective switch protection, and 3) selective protection by the Relocation Page Index File. The protection offered by the optional Page Index File is discussed in Section 5.

If any attempt is made to Write at a protected address during Non-Executive mode, the illegally addressed location remains unaltered (Write is changed to a Read), the console ILLEGAL WRITE indicator lights, and program execution continues. The Illegal Write condition is recorded by setting bit 05 of the internal status sensing network.

During Executive mode, a protected address remains unaltered (Write is changed to a Read) during all write operations except those occurring in Monitor state and during Block Control operations. The condition is not recorded on the status line. Refer to the Illegal Write Interrupt discussion in Section 4 for additional information.

The addresses used by the interrupt system for storing the (P) and the interrupt code are never protected during an interrupt sequence. An interrupt address bias can be selected for a Central Processor, enabling four processors to access the same physical bank zero, and still have different interrupt addresses (see Section 4).

Permanent Protection

Auto Load/Auto Dump program areas are permanently protected against alteration by the hardware. During Non-Executive mode operations the protected addresses are 77740 through 77777. During Executive mode operation, locations 03700 through 03777 are protected when referenced through Page Index Zero. Protection of these areas is disabled only when the ENTER AUTO PROGRAM switch on the console is pressed.

Selective Protection

Two separate areas within a designated 32K of storage can be protected during Non-Executive or Executive mode by using Storage Protect switches (UPPER WRITE LIMIT and LOWER WRITE LIMIT) located on the Computer Maintenance Panel (see Section 7). With this feature, protection can be given to the resident monitor program, and to another program that may be operating. In Executive mode, the switches apply only to areas referenced through the lower 16 indexes (State 0) of the Page Index File.

An area increasing in address from address 00000 may be protected in multiples of 512_{10} locations. The area decreasing from address 77777 can similarly be protected. The number of locations protected in an area is determined by setting six toggle switches associated with that area; each of the 77_8 possible settings represents one multiple of 512 locations. The UP and DOWN switch positions signify "1" and "0", respectively. Figure 2-2 illustrates the protection scheme. See Table 2-1 for sample switch settings and their corresponding protected areas.

All switch settings are disabled by pressing the DISABLE STO PROTECT switch on the console.

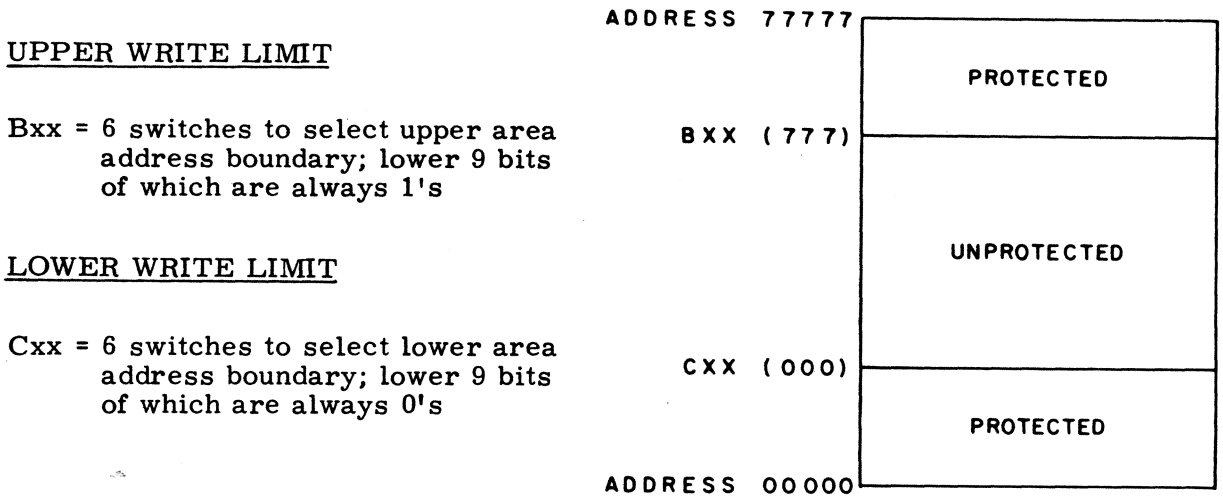


Figure 2-2. Storage Protect Scheme

TABLE 2-1. SAMPLE STORAGE PROTECT SETTINGS

Bxx SETTING	LOCATIONS PROTECTED (UPPER AND LOWER AREAS)	Cxx SETTING
76	01000_8 = 512	01
75	02000_8 = 1,024	02
74	03000_8 = 1,536	03
67	10000_8 = 4,096	10
57	20000_8 = 8,192	20
40	37000_8 = 15,872	37
37	40000_8 = 16,384	40
36	41000_8 = 16,896	41

With Standard Option 10288 main core storage is protected against write references to wrong addresses, caused by a file failure, that is detected as a parity error. I/O, S/M, and type buffers are aborted after current memory request, upon detection of associated page file parity error.

3. INPUT/OUTPUT SYSTEM

GENERAL INFORMATION

Communication channels provide a buffer between the computation section and peripheral controllers in a system, thus preventing a tie-up of the computation section while awaiting a response from an external equipment. Block Control regulates all I/O buffers; however, the operations must be initiated by the main program through the Main Control section. Prior to actual data exchange, the program must execute instructions which connect the equipment to the channel, specify operating conditions, and initiate the Read or Write operation. Once the Read or Write is initiated, a communication channel exchanges data between the peripheral unit and core storage independent of the Main Control section.

In the 3500 System, data is transferred between the computer and its associated peripheral equipment through a CONTROL DATA 3507-1 Communication Channel. The channel can operate in 12-/24-bit mode to perform 12-bit data transfers with assembly/disassembly or high-speed 24-bit data transfers. The operating mode is chosen, as application requires, by a simple switch selection and the proper I/O interface cabling. A maximum of eight I/O channels may be incorporated into a single system, any combination of which may be in 12-/24-bit mode. The channel is bi-directional and can communicate with any of eight peripheral controllers. A controller, in turn, may control up to eight peripheral units or devices. However, at any given instant, one channel may communicate with only one unit. Figure 3-1 illustrates the 3500 I/O system.

For purposes of channel selection and identification, the eight possible channels are designated by the numbers 0 through 7. This number is referred to as the designator 'ch' during channel programming.

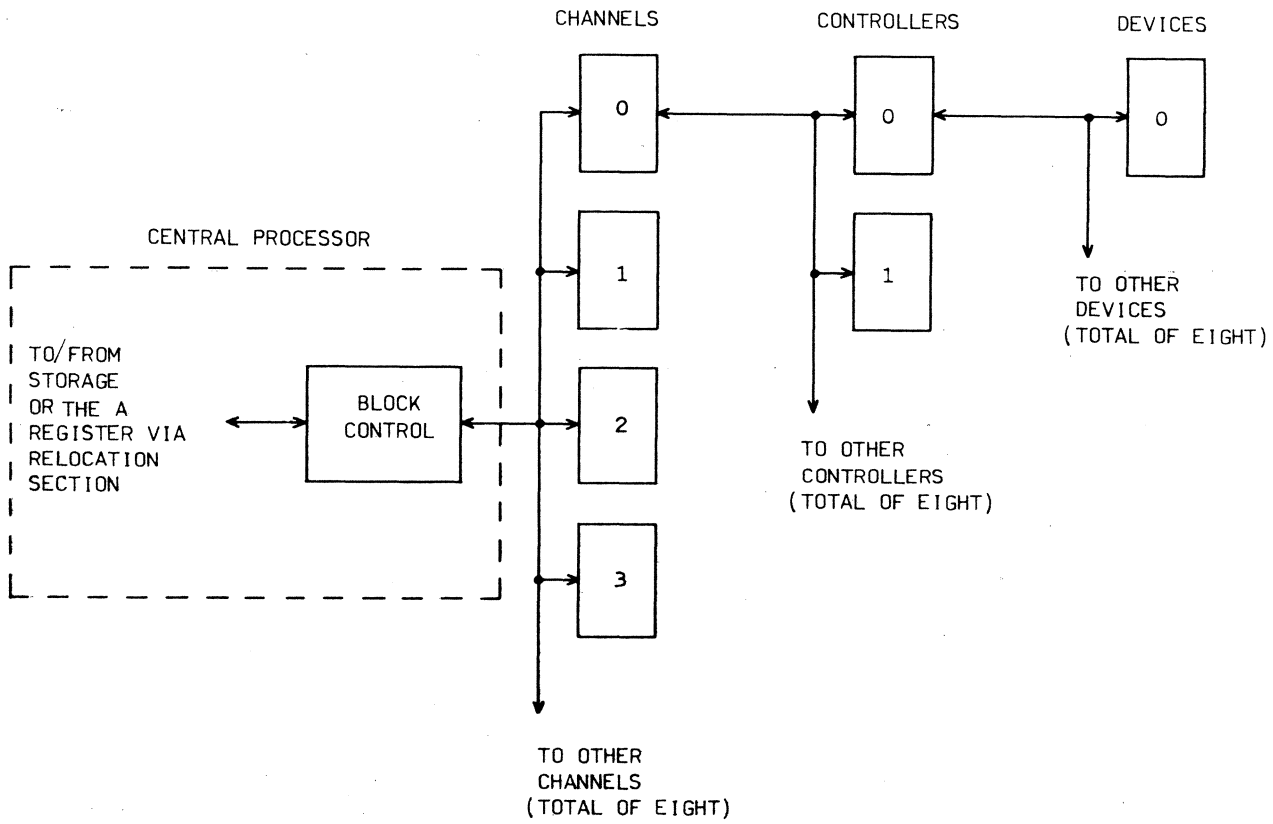


Figure 3-1. 3500 I/O System

CHANNEL OPERATING MODES

12-Bit Mode

In 12-bit mode, the channel is an assembly/disassembly data channel with a 26-bit interface to Block Control and a 13-bit interface to external equipment, including parity bits. The assembly/disassembly logic included within the channel eliminates a memory reference during the following I/O operations:

12- to 24-bit assembly on word-addressed input to storage.
24- to 12-bit disassembly on word-addressed output from storage.

During assembly, the channel receives two 12-bit bytes from external equipment and assembles them into a 24-bit word before storing them in memory. For disassembly, the channel accepts a 24-bit word from storage and transmits it to the equipment in two 12-bit bytes.

The channel accepts 6-bit characters during character-addressed I/O, with Block Control and storage performing all assembly/disassembly operations.

24-Bit Mode

In 24-bit mode, the data channel has a 26-bit interface to Block Control and peripheral equipment, including parity bits. This arrangement is especially suitable for straight word-addressed I/O programming with high-speed 24-bit devices. Block Control and storage perform all assembly/disassembly operations programmed in 24-bit mode. If assembly/disassembly is not specified and 24-bit mode is used with a 12-bit device, the upper half of consecutive storage words are filled with zeros during input; the upper 12 bits are lost during output.

The interface signals to external equipment and the parity checking employed in conjunction with both channel modes are discussed later in this section. Refer to the individual I/O instructions in Section 6 for additional programming information.

INTERFACE SIGNALS

Figure 3-2 shows the main signals exchanged between the communications channel and peripheral controllers. The bi-directional data lines carry 12 or 24-bit data words during both input and output operations. These lines also transmit Connect and Function codes to the controllers.

The 12 status lines carry status information from the selected peripheral controller. Each line is associated with specific status condition such as READY or BUSY. The status lines between a channel and controller are active only if the controller has been connected by a CON (77.0) instruction, or if a Connect was attempted and a Reject was returned by the controller. In the latter case, the status lines remain active until another Connect is attempted, so that the programmer can sense status to determine the reason for the Reject.

The interface includes eight external interrupt lines, one from each of the eight possible controllers. Certain controllers need not be connected to return an interrupt signal to the data channel. External interrupt lines, designated 0-7, correspond to the Equipment Number switch setting in each controller. For a complete description of the I/O interface signals, as well as timing information, refer to the 3000 Series Input/Output Specifications Manual, Pub. No. 60048800.

In addition to the data, status, and interrupt lines, there are a number of control signals that coordinate the exchange of information between a channel and associated peripheral controllers.

INPUT/OUTPUT TRANSMISSION PARITY

Parity is checked on all information exchanged between the computer and external equipment; odd parity is always used. Block Control, in conjunction with the channel and external equipment, checks parity on all external transfers. The Relocation section checks parity on addresses and data for storage references during I/O operations (see Section 2). A Parity Error interrupt may result from a parity error during a storage reference, but not because of external transmission parity errors.

Extraneous I/O transmission parity errors may result during an input operation with a controller that sends more data than requested. For example, an extraneous error may occur if character addressed input, with 6- to 24-bit assembly, is specified and the controller sends 12 or 24 bits to the channel.

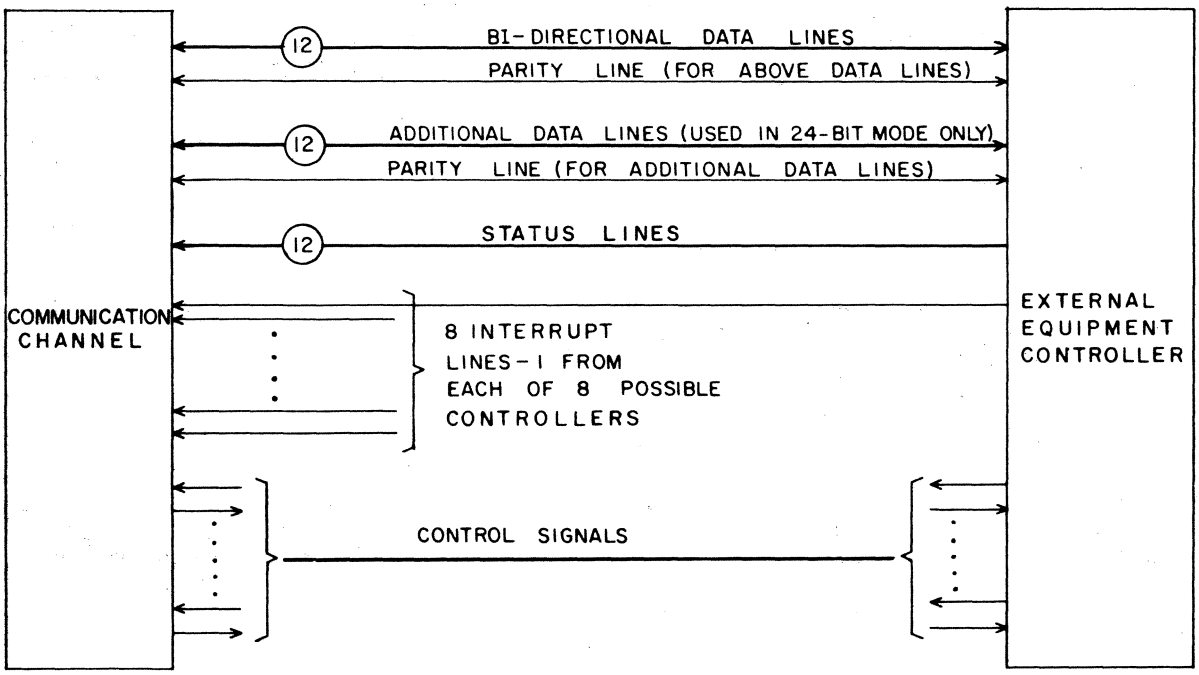


Figure 3-2. I/O Interface Signals

For details on external equipment parity and equipment responses to parity error signals, see the 3000 Series Peripheral Equipment Reference Manual, Pub. No. 60108800.

Parity Checking During Input

During an input (Read) operation the external equipment generates and sends 1 parity bit to the channel for each 12-bit byte of data (2 bits are sent for a 24-bit device). The data channel retains the parity bit(s) while the data is forwarded to the D_4 register in Block Control. Block Control generates new parity for the data and sends the parity bit(s) back to the channel where it is compared with the parity received from the

external equipment. If a parity error exists, the error is recorded in the channel by placing a "1" on Internal Status Line 0. The input continues despite the external parity error, and if the error is not sensed before another operation is initiated on the same channel, the condition is lost to the programmer.

The channel parity error logic is cleared if a Connect, Function, Write, or another Read operation is initiated on the channel. The error may also be cleared by channel clear instructions.

Parity Checking During Output

During an output (Write) operation, Block Control generates 1 parity bit per 12 bits of data received from storage or the A register. The parity bit(s) and data are sent to the channel for transmission to the equipment. The external equipment checks parity on the data received and transmits an External Parity Error signal to the channel if an error is detected. The channel records the condition by placing a "1" on Internal Status Line 0.

The output operation continues despite the external parity error and if the error is not sensed before another operation is initiated on the same channel,the condition is lost to the programmer. The error condition in the channel is cleared upon initiation of a Connect, Function, Read, or Write operation, or by a channel clear instruction. However, to assure clearing the source of the error signal, the external equipment should be cleared.

Parity Checking During Connect

Parity is checked during a Connect operation in the same manner as during a Write. However, the data channel does not receive a Parity Error signal from the controller if a parity error occurs, and the Central Processor generates an internal reject after 100 usec. The following normally occurs when a transmission parity error is detected during a Connect:

- The equipment does not connect
- Neither a Reply nor Reject are returned
- An indicator lights on the controller

Parity Checking During Function

Parity is checked during a Function operation in the same manner as during a Write. The external equipment returns an External Parity Error signal if a parity error is detected, and a "1" is placed on Internal Status Line 0. The Central Processor generates an internal reject after 100 usec since neither a Reply nor Reject are returned to the channel. The device does not perform the function and the condition should be cleared before continuing. If cleared, the device must be reconnected before the function can be accomplished.

EXECUTIVE MODE INPUT/OUTPUT

An Executive interrupt is generated when an I/O instruction is read from memory during Program state of Executive mode. The interrupt automatically returns the computer to the Monitor state, thereby enabling the monitor program to perform the I/O task. The monitor minimizes the idle time of the peripheral equipment by assigning channels and devices as they become available. I/O can be programmed for the same channel (preferably Channel 0) in several programs since the monitor modifies the selection as necessary by OR ing the contents of the Channel Index register (CIR) with the 'ch' designator in the instruction prior to initiating the operation (see 77.54, A to Channel Index Register, instruction).

A maximum data block of 32K words can be transferred by a single instruction; the state number (contents of ISR or OSR) is appended to the basic address in the Register File during I/O with storage.

AUTO LOAD/AUTO DUMP

The Auto Load and Auto Dump feature of the computer allows the programmer two groups of continuous storage locations for storing frequently used subroutines. These subroutines may be used for any purpose, but normally are used for calling in a particular tape or other device, or to initiate a standard output.

By pressing the AUTO LOAD switch on the console when the computer is in Non-Executive mode, the computer automatically jumps to address 77740 and executes the instruction stored there. The Auto Load routine is allotted 16 addresses (77740 through 77757). The last instruction in this routine is normally a jump to the first address of the program to be executed.

Pressing the AUTO DUMP switch under the same conditions stated for Auto Load causes the computer to jump to address 77760 and execute the instruction stored there. Sixteen addresses (77760 through 77777) are reserved for the Auto Dump routine.

When the computer is operating in Executive mode, the Auto Load routine is stored in 32 locations encompassing addresses 03700 through 03737. Likewise, the Auto Dump routine is stored in 32 locations ranging from address 03740 through 03777.

Hardware protection given the Auto Load/Auto Dump areas is described in Section 2.

TRANSMISSION RATES

In obtaining the 3500 I/O transmission rates, a variable-speed channel exerciser was used. The exerciser measured rates by indicating a Lost Data condition when the speed of the exerciser exceeded the speed of the channel. The tests were performed under the following conditions:

1. Cable lengths of 40 feet (at 2 nsec/foot) and controller turn-around time of 300 nsec were used, for a total overhead of approximately 450 nsec.
2. Fully bufferred output operations were used, i.e., no restrictions were placed on the program or Real-Time Clock.
3. The main program was designed to create maximum conflicts with the output operations within Block Control, Relocation, and storage.

Table 3-1 lists the worst case times for consecutive character, byte, and word transfers. For 24 to 12-bit disassembly operations, where a word is read from storage but is transmitted to the equipment in two separate bytes, the worst case time for the two bytes can be determined by adding 0.750 usec to the time listed in Table 3-1.

The number of channels varied in each test case. The second time listed under each channel heading is the applicable transfer time with a buffered Move (72) instruction simultaneously in progress.

TABLE 3-1. TRANSMISSION RATES*

	CHANNEL 0	CHANNEL 1	CHANNEL 2	CHANNEL 3
TEST #1	2.8 4.1	N/A	N/A	N/A
TEST #2	3.7 4.3	3.7 5.0	N/A	N/A
TEST #3	4.3 4.3	4.5 5.2	4.6 6.0	N/A
TEST #4	4.3 4.3	5.1 5.2	5.3 6.0	5.4 6.8

*All times listed are in usec.

PARITY CHECKING IN REGISTER FILE

Register File parity hardware (some models) characteristics are as follows:

- Generate and check 27-bit parity.
- Parity is generated and written when writing into the file. (IRT write, WRF mode, write 0X, IX during I/O activate, write 2X, 3X during S/M activate, write 0X, 2X, 3X during buffer cycles.)
- Parity is checked when reading the file. (IRT read, RRF mode, read 0X, 1X during I/O buffer cycle, read 2X, 3X during S/M, RTC, type buffer cycle.)
- Wrong parity will remain until new word with correct parity is rewritten.
- Parity error code is 7XX where XX equals register file address.
- Block write into memory if RF parity error occurs on block control write storage request.
- Abort I/O, S/M and Type buffers, after current memory request, upon detection of associated RF parity error.
- Enable parity checking only if the PARITY STOP switch or the PARITY INTERRUPT switch is on.

4. INTERRUPT SYSTEM

GENERAL INFORMATION

The 3500 interrupt system automatically checks for the presence of certain internal and external interrupt conditions without having these tests in the main program. These interrupts are classified into four groups.

1. Normal interrupts include those interrupts considered most likely to occur, or those that are generally used by the programmer to control a program. These are: I/O, Internal Condition, Manual, Associated Processor, and Executive interrupts.
2. Abnormal interrupts are not fully controlled by the programmer; the interrupts provide program protection and enable efficient memory allocation. Included are Storage Parity Error, Power Failure, and Illegal Write interrupts. Some models add Page File Parity Error, Register File Parity Error, and undefined condition.
3. Trapped Instruction interrupts in the 3500 are generated when either the optional Business Data Processing section has not been enabled and execution of a BDP instruction is attempted, or the FP MODE switch is OFF and a FP instruction is attempted.
4. Real-time interrupts are generated by the optional Real-Time Interrupt Processor. The RIP improves the 3500 interrupt system performance by providing up to 64 additional priority interrupt lines and decreasing interrupt response time. The RIP is under program control.

The hardware automatically tests for interrupt conditions at different times during instruction execution. If an interrupt condition is recognized, execution of the main program halts. The contents of the P register and an interrupt code are stored, and an interrupt routine is entered. The previously stored interrupt routine performs the necessary functions for the existing condition and then jumps back to the last unexecuted step in the main program. The instruction in progress when the interrupt was recognized is executed when the main program is resumed.

Special considerations must be given to handling interrupts that occur during Executive Mode operations. Refer to the discussion labeled Interrupts During Executive Mode for details. Status sensing is also described in this section.

NORMAL INTERRUPTS

Input/Output Interrupts

Input/Output interrupts can be selectively enabled or disabled by setting or clearing bits in the Interrupt Mask register.

I/O Channel Interrupts

Any of the eight possible I/O channels may be programmed to generate an interrupt condition upon completion of an operation. The condition is selected in the I/O instruction (bit 17 of P + 1) and occurs when the end of an input or output block is reached. The end of a block is reached when a complete pass is made through the specified address range or an End of Record signal (unless using word count control) is received from an external equipment during input.

I/O Equipment (External) Interrupt

An interrupt condition in a connected external equipment (peripheral controller) is selected by issuing an appropriate function code to the equipment (see SEL instruction). The standard interrupt conditions that may be selected are: Interrupt on End of Operation, Interrupt on Abnormal End of Operation, and Interrupt on Ready and Not Busy. The interrupt signal may be received from any of the eight peripheral controllers that may be serviced by any of the eight possible I/O channels (maximum of 64 interrupt lines).

All external interrupts are locked out for 1 μsec after the execution of an INCL (77.50) or IOCL (77.51) instruction to allow for signal propagation time, thus preventing an interrupt from being detected a second time. Interrupt selections remain set in the equipment until released or cleared and may cause an interrupt even if the equipment is connected to another peripheral unit (refer to Interrupt Clearing later in this section).

Internal Condition Interrupts

Any of six internal conditions may cause an interrupt during the execution of a program. Internal condition interrupts are masked and require that the interrupt system be enabled.

Arithmetic Overflow Fault

The Arithmetic Overflow fault is set when the capacity of the adder is exceeded. Overflow is not sensed for instructions 53. (0-3)4, [Transfer (A)+(B^b) to A].

Single Precision: The capacity is exceeded in instructions 15.(4-7), 30, 31, and 34 if $|\text{Result}| \geq 2^{23}$.

Double Precision: The capacity is exceeded in instructions 32 and 33 if $|\text{Result}| \geq 2^{47}$.

Divide Fault

The Divide fault sets if |quotient| $\geq 2^{23}$, and 2^{47} for 24-bit precision and 48-bit precision, respectively. Therefore, attempts to divide by too small a number, including positive and negative zero, result in the Divide fault.

A Divide fault also occurs when a floating point divisor is either equal to zero or is not in floating point format. The results in the A, Q, and E registers are insignificant if a fault occurs.

Exponent Overflow/Underflow Fault

During all floating point arithmetic operations, the exponent fault is set if the exponent of the result exceeds $+1777_8$ or is less than -1777_8. The fault is also set if the SFPF (77.71) instruction is executed.

BCD Fault

A BCD fault is generated by the optional BDP section if:

1. The lower 4 bits of any character in field A (except the sign character) exceeds 11_8 during a numeric character operation
2. The lower 4 bits of the sign character in field A exceed 12_8 during a numeric character operation
3. The upper 2 bits of any character in field A (except the sign character) do not equal '00' during a numeric character operation
4. An arithmetic carry out of the highest order character of field C occurs during an ADM or SBM instruction
5. Field A length ($\$_1$) exceeds field C length ($\$_2$) for an ADM or SBM instruction
6. Incorrect field length specified (FRMT instruction):
   a) The two fields are not aligned, $\$_1 = 2$ and $\$_2 = 6$
   b) Illegal count used, $\$_2 \neq 3, 6, 12_8, 16_8, 22_8$, etc.
7. A carry occurs out of the 14th character position during a CVBD instruction.
8. A field ($\$_1$) of more than 14 BCD characters is specified during a CVDB instruction
9. The SBCD (77.72) instruction is executed

Search/Move Interrupt

Search/Move control may be programmed to generate an interrupt condition upon completion of a 71 or 72 instruction. The condition is selected in the Search or Move instruction (bit 17 of P) and occurs upon:

1. Completion (address range searched or search character located) of an equality or inequality search instruction (SRCE or SRCN)

2. Completion of a block move instruction (MOVE)

Real-Time Clock Interrupt

A Real-Time Clock Interrupt condition is generated when the current clock time (register 22) equals the time specified by the clock mask (register 32).

Executive Interrupt

The Executive interrupt is not masked and is always recognized when the condition is generated; the interrupt system need not be enabled. Any attempt to execute one of the following instructions during Program state of Executive mode generates an Executive interrupt.

1. Halt instruction (00.0)
2. Inter-register transfer (instructions 53.(4-7) (1-3)) into Register File locations 00 through 37
3. Instructions with octal codes 71 through 77, except the 77.71 (SFPF) and 77.72 (SBCD) instructions

When the Executive interrupt has been processed, the computer reverts to the Monitor state, and any of the instructions in the three categories above can be executed. The interrupt is automatically cleared when recognized.

Manual Interrupt

The Central Processor may be manually interrupted during execution by pressing the MANUAL INTERRUPT switch on the operator's console. Although the interrupt is not masked, the interrupt system must be enabled for the condition to be recognized. Once set, the condition remains active until recognized, or until Master Clear.

Associated Processor Interrupt

In a system of two or more central processors, each processor may interrupt one other processor in the system by executing an IAPR (77.57) instruction. However, the interrupt system must be enabled in the processor being interrupted. Likewise, a processor may be interrupted by only one other processor. This interrupt is not masked and once set, remains active until recognized, or until Master Clear.

Normal Interrupt Control

Enabling and Disabling Interrupts

The interrupt system must be enabled to recognize all normal interrupts except Executive interrupt. The interrupt system is enabled by executing the EINT (77.74) instruction. The DINT (77.73) instruction disables the interrupt system. After an interrupt condition is recognized and an interrupt subroutine is entered, all other

normal interrupts are disabled. When leaving the interrupt subroutine the interrupt system must again be enabled by executing the EINT instruction if normal interrupt conditions are to be recognized. Special considerations for Executive Mode interrupt handling are given later in this section. Refer to descriptions of EINT and DINT instructions in Section 6 for special conditions regarding the actual interruption of the Central Processor.

Interrupt Mask Register

The programmer can choose to honor or ignore normal interrupt conditions by means of the 12-bit Interrupt Mask register. All but three of the normal interrupt conditions require a representative bit to be set in the mask register in order to be recognized. The Associated Processor, Executive, and Manual interrupts are not masked. Table 4-1 lists the Interrupt Mask register bit assignments.

The mask is selectively set with the SSIM (77.52) instruction and selectively cleared by the SCIM (77.53) instruction. The contents of the register may be transferred to the upper 12 bits of the A register with the COPY (77.2) or CINS (77.3) instruction.

TABLE 4-1. INTERRUPT MASK REGISTER BIT ASSIGNMENTS

MASK BITS	MASK CODES	INTERRUPT CONDITIONS REPRESENTED
00	0001	I/O Channel 0
01	0002	1
02	0004	2 (Includes interrupts
03	0010	3 generated within the
04	0020	4 channel and external
05	0040	5 equipment interrupts.)
06	0100	6
07	0200	7
08	0400	Real-time clock
09	1000	Exponent overflow/underflow and BCD faults.
10	2000	Arithmetic overflow and divide faults
11	4000	Search/Move completion

Interrupt Sensing

The programmer may selectively sense active interrupt conditions by using the INTS (77.4) and INS (77.3) instructions. The interrupt system need not be enabled for sensing. If the selected interrupt condition is present, a RNI is performed at P + 1; if not, a RNI occurs at P + 2. Refer to the instruction descriptions for the required sensing masks.

Sensing the presence of internal fault conditions automatically clears them. Interrupt lines representing channels not present in the system are sensed as being active.

Interrupt Clearing

I/O channel interrupts are cleared by:

- Executing an INCL (77.50), IOCL (77.51), or CLCA (77.512) instruction, or

- Pressing MASTER CLEAR or EXTERNAL CLEAR.

I/O equipment interrupts* are cleared by:

- Executing an IOCL (77.51) instruction,
- Reselecting or releasing the interrupt with a SEL (77.1) instruction, or
- Pressing MASTER CLEAR or EXTERNAL CLEAR.

Internal condition interrupts are cleared by:

- Sensing with an INTS (77.4) or INS (77.3), after which interrupt conditions are automatically cleared,
- Executing an INCL (77.50) instruction,
- Executing an IOCL (77.51) instruction - clears only Search/Move interrupt, or
- Pressing MASTER CLEAR or INTERNAL CLEAR.

The Manual, Executive, and Associated Processor interrupts are automatically cleared when recognized, or when master cleared.

Interrupt Address Bias

The Central Processor is assigned a unique number (0-3) for biasing addresses assigned for interrupt processing. Two binary toggle switches, located on the end connector panel, are set to bias bits 6 and 7 of interrupt addresses for a processor. This permits systems with up to four processors to share the same physical bank zero of memory, and still have different interrupt addresses.

Interrupt addresses presented in this manual include no bias; all addresses are for Processor 0.

Normal Interrupt Recognition

Three conditions must be met before a normal interrupt can be recognized:

1. A bit representing the interrupt condition must be set in the Interrupt Mask register (except for Associated Processor, Executive, and Manual interrupt).
2. The interrupt system must be enabled (except for Executive interrupt), and
3. The interrupt condition(s) must exist.

The computer tests for interrupt conditions at five different times. A standard test is made at the end of each Read Next instruction (RNI) and Indirect Address (RADR) sequence. Special tests are made at the end of the Read Operand (ROP) sequence for a MEQ (06) and MTH (07) instruction, during the delay portion of a PAUS (77.60) and PRP, and during BDP instructions.

*Refer to instruction description of INCL for special case involving external equipment interrupts.

If two or more selected interrupt conditions are active during an interrupt test, interrupt recognition occurs according to the priority given in Table 4-5. When a normal interrupt is recognized, the following occur:

1. The address of the current unexecuted instruction (P) is stored in the lower 15 bits of address 00004.
2. An interrupt code representing the recognized interrupt condition is stored in the lower 12 bits of address 00005. Refer to Table 4-2 for the octal codes stored for the different normal interrupts.
3. Program control is transferred to address 00005 and a RNI sequence is initiated.

TABLE 4-2. NORMAL INTERRUPT CODES

CONDITIONS	CODES
External interrupt	*00LCh
I/O Channel interrupt	010Ch
Real-Time Clock interrupt	0110
Arithmetic Overflow fault	0111
Divide fault	0112
Exponent Overflow fault	0113
BCD fault	0114
Search/Move interrupt	0115
Manual interrupt	0116
Associated Processor interrupt	0117
Executive interrupt	0120
Restore interrupt	**0317

*L = line 0-7 and Ch = channel designator, 0-7
**This code can occur only if the optional RIP is present

ABNORMAL INTERRUPTS

Tests for abnormal interrupts are made at the same time as for normal interrupts (see page 4-6). Abnormal interrupts are not selected by the programmer and are always recognized when the interrupt conditions exist. The interrupt system need not be enabled; however, if it is enabled, the abnormal interrupts will disable it. These interrupts have priority over all other interrupts and rank within their own group as follows: Storage Parity Error, Illegal Write, and Powerfail. A complete priority listing is given in Table 4-5.

Storage Parity Error/No Response Interrupt

A Storage Parity Error interrupt has the highest priority of all interrupts and can occur if either a storage parity error is detected, or if a storage module does not respond when referenced. The condition is not generated during the operand cycle (ROP) of the Test Memory Availability instruction, an instance where no response is anticipated.

A Storage Parity Error interrupt can also occur if a RIP Parity Error is detected while restoring the operational registers following a Real-Time interrupt. A RIP Parity Error can occur only on the instruction immediately following the Restore instruction. Refer to the discussion of Real-Time Interrupts in this section.

The PARITY INTERRUPT switch on the console must be active for the interrupt to occur. If the PARITY STOP switch is active the computer stops when a Storage Parity Error, RIP Parity Error, or No Response condition is detected. The two switches cannot be simultaneously active; pressing the PARITY STOP switch overrides the PARITY INTERRUPT condition.

If the Block Control section has storage priority at the time of interrupt, the address of the next instruction to be executed is stored in the lower 15 bits of location 00020. The appropriate register file location* contains the approximate address where the error occurred. An interrupt when Main Control has priority causes the address of the current instruction to be stored in location 00020. If the error condition is detected during any of the RNI's for the multiple word instructions (BDP or I/O) or as a result of the RIP Parity Error, the (P) is stored at location 00020.

A code representing conditions within the processor at the time of interrupt is automatically stored in the lower 12 bits of location 00021. A RNI is then performed at location 00021. The stored address and code enable the interrupt routine to isolate the storage area where the error occurred, and also aid in program recovery. Table 4-3 lists the various codes and their interpretations.

The instruction in progress when the interrupt is detected may be executed although the results are not necessarily correct. Once the Parity Error or No-Response condition is detected, additional errors are not recognized until a DINT (77.73) instruction is executed.

TABLE 4-3. PARITY ERROR INTERRUPT CODES

REASON FOR INTERRUPT	TYPE OF OPERATION OR SEQUENCE IN PROGRESS	CODE	
No-Response	Block Control - (73-76)	00X0	X = ch
Parity Error	Block Control - (73-76)	**00X2	X = ch
No-Response	Block Control - 71, 72, or Typewriter I/O	01X0	(X = 0, Srch), (X = 1, Move), (X = 3, TWR)
Parity Error	Block Control - 71, 72, or Typewriter I/O	**01X2	(X = 0, Srch), (X = 1, Move), (X = 3, TWR)
No-Response	Main Control - RNI or RADR	00X1	(X = 0, RNI), (X = 2, RADR)
Parity Error	Main Control - RNI or RADR	**00X3	(X = 0, RNI), (X = 2, RADR)
No-Response	Main Control - ROP or STO	0005	
Parity Error	Main Control - ROP or STO	**0007	
PF Parity Error	Main Control - RNI, RAD, STO or Block Control I/O reading PF	0600	
PF Parity Error	Main Control - APF or Enter PF writing PF	0601	
RF Parity Error	IRT, RRF, I/O Bfr, S/M, RTC, Type Bfr Reading RF	07XX	(XX = RF Location)
Undefined Condition	Simultaneous PE's	1XXX	

** Bits 7 and 8 of this code are translated as follows:

* The register file contains the current address for Block Control operations (I/O, Search and Move). Refer to the individual instruction descriptions (in Section 6) for the register file location associated with each instruction.

11 10 9 8 7 6 5 4 3 2 1 0 Bits

0 0 0 = Data Parity Error or No Response
0 0 1 = RIP Parity Error
0 1 0 = Address Parity Error
0 1 1 = RF or PF Parity Error
1 X X = Undefined Condition

Illegal Write Interrupt

This interrupt has priority over all other interrupts except the Storage Parity Error interrupt. When the interrupt condition is recognized, the interrupt system is disabled, the (P) are automatically stored at address 00014, and the next instruction is read from address 00015. The instruction at P is not executed (see the 06, 07, and BDP instructions for exceptions).

The Illegal Write interrupt occurs only during Program state of Executive mode. The interrupt is disabled during Monitor state, Search/Move and I/O buffer cycles, keyboard operations, and during Non-Executive mode.

The Relocation section compares incoming addresses with protected addresses and checks the Page Index that is referenced for Illegal Write Interrupt conditions.

An Illegal Write interrupt occurs during the following:

1. A Write operation into an area protected by the Storage Protect switches.
2. A Write operation into the Executive Auto Load/Auto Dump area when referenced through Page Index Zero.
*3. A Read or Write operation when bits 9 and 10 of the original address specify a quarter page equal to or greater than PL, when PL $\neq$ 0.
*4. A Read operation if the E designator for any referenced index equals "1" and PA, PL, and PP are equal to 0.
*5. A Write operation if the E designator for any referenced index equals "1".
*6. On the first operand reference of a Double Precision instruction if the second operand reference satisfies condition 3, 4, or 5 as listed above.

Power Failure Interrupt

If source power to the computer system fails, the power failure is detected and the computer program is interrupted. This interrupt is necessary to prepare a controlled shutdown and to prevent data loss. The operation requires 16 ms for detection, and allows up to 4 ms for processing the special Power Failure interrupt routine.

When a Power Failure interrupt occurs, the computer stores the (P) in the lower 15 bits of address 00002 (upper 9 bits remain unaltered) and performs a RNI from address 00003.

The interrupt system is disabled during a power failure sequence; that is, the hardware simulates the execution of a DINT (77.73) instruction.

---

* This Illegal Write interrupt condition occurs only if the optional Relocation Page File portion of the Relocation section is present.

TRAPPED INSTRUCTION INTERRUPTS

A special trapped instruction sequence is entered if an attempt is made to execute one of the following instructions:

1. Any of the 3500 BDP instructions when the BDP MODE switch is OFF or the BDP is not present.
2. Double precision multiplication/division and all Floating Point instructions when the FP MODE switch* is OFF.

During real-time applications, interrupt response time is reduced if the FP MODE switch is OFF and Floating Point instructions are trapped.

Trapping the 3500 BDP instructions allows the BCD instructions used in the 3100/3200 Systems to be trapped, maintaining upwards compatability by software execution.

The interrupt system is not disabled when a trapped instruction is detected. Table 4-4 lists the 3500 instructions that are trapped.

TABLE 4-4. TRAPPED INSTRUCTIONS

BDP MODE INACTIVE	
64.0	MVE and MVE, dc
64.1	MVBF
64.2	MVZF
64.3	MVZS and MVZS, dc
64.4	FRMT
64.4	EDIT
65.0	SCAN, LR, EQ and SCAN, LR, EQ, dc
65.2	SCAN, LR, NE and SCAN, LR, NE, dc
65.1	SCAN, RL, EQ and SCAN, RL, EQ, dc
65.3	SCAN, RL, NE and SCAN, RL, NE, dc
66.0	CVDB
66.1	CVBD
66.2	DTA and DTA, dc
66.3	ATD and ATD, dc
66.4	PAK
66.5	UPAK
67.0	ADM
67.1	SBM
67.2	ZADM
67.3	CMP and CMP, dc
67.4	TST and TSTN
70.0	JMP, HI
70.1	JMP, ZRO
70.2	JMP, LOW
70.6	LBR**
70.7	SBR**
FP MODE INACTIVE	
56	MUAQ, I
57	DVAQ, I
60	FAD, I
61	FSB, I
62	FMU, I
63	FDV, I

** When the BDP MODE switch is OFF, the LBR and SBR instructions are trapped if in Non-Executive mode or Program state. These instructions are No-Ops if in Monitor state.

* 3514 CPU's only. In 3504-1 FP instructions cannot be disabled.

All normal, abnormal, and real-time interrupts have priority over the trapped instruction interrupts.

The following operations take place when a trapped instruction is recognized:

1. The address of the next sequential program step, P+1, is stored in the lower 15 bits of address 00010.
2. The upper 6 bits of the instruction in the F register are stored in the lower 6 bits of address 00011. The upper 18 bits remain unchanged.
3. Program execution commences at address 00011.

EXAMPLE: Execution of a MVE (64. 0) instruction attempted with BDP MODE switch not active.

Address P	- 54430	64 0 30345	Address 00010	01 0 54431
Address P+1	- 54431	00 0 20267	Address 00011	14 0 00064
	-----	-- - -----	-----	-- - -----

REAL-TIME INTERRUPTS

Tests for real-time interrupts are made at the same time as for the normal interrupts. Real-time interrupts require that the optional Real-Time Interrupt Processor is present and enabled.

The RIP can receive up to 64 interrupt signals from external sources. These interrupts have priority over normal CPU interrupts, but are lower in priority than abnormal CPU interrupts. Any desired priority can be attained by cabling. Line 00 has the highest priority and line 63 the lowest.

Enable/Disable RIP Interrupts

Individual interrupt lines are enabled by corresponding interrupt mask bits; mask bits are set by the SRIM (16. 01) instruction and cleared by the CRIM (16. 00) instruction. The Mask register is not cleared by a Clear function. The RIP is active only if enabled. The ERIP (16. 05) instruction enables the RIP and the DRIP (16. 04) instruction disables the RIP; these instructions do not affect the enabling or disabling of CPU interrupts, and likewise,the EINT and DINT instructions do not affect the RIP.

The RIP normally remains enabled after interrupting to allow processing of real-time interrupts as fast as they are received in the RIP and processed by the CPU. However, the programmer has the option of selecting an automatic disable which disables the RIP following any real-time interrupt. The automatic disable feature is selected by the SAD (16. 06) instruction, deselected by the DAD (16. 07) instruction, and cleared via any Clear function. If the RIP is disabled for any reason, it can be re-enabled only by an ERIP instruction.

Because CPU abnormal interrupts have a higher priority than any RIP interrupt, the recognition of an abnormal interrupt by the CPU automatically disables the RIP. The first ERIP instruction following an abnormal interrupt enables the RIP if the RIP was enabled at the time of the abnormal interrupt; if the RIP was not enabled, the RIP ignores the first ERIP instruction. All successive ERIP instructions are executed in the normal manner, until another abnormal interrupt occurs.

The RIP always records interrupts and, if enabled, recognizes the highest priority interrupt recorded.

NOTE

To record an interrupt is to remember an interrupt signal received in the RIP; to recognize an interrupt is to pick the highest priority recorded interrupt and send it to the CPU.

If the RIP is not enabled, it continues to record interrupts but does not process them. This allows recognition of the highest priority recorded interrupt when the RIP is enabled. In addition, whenever the RIP is disabled by a DRIP instruction, a Master Clear, or by an abnormal interrupt, the interrupt line currently recognized reverts to the recorded state. After interrupting on lines 20_8 - 77_8, the recognition of interrupts is blocked until a CRA instruction is executed. This instruction indicates the end of register saving by program control and furthermore, it must be executed following any interrupt on lines 20_8 - 77_8 to re-enable RIP interrupt recognition and allow proper priority to be maintained.

Interrupt Addressing

Upon recognition of an interrupt by the RIP, a 15-bit interrupt address (Figure 4-1) is sent to the CPU which specifies where program control is to be transferred. The interrupt address is a composite of a base address and the recognized interrupt line. The base address is set up by executing the SIBA (16.02) instruction and remains set until another SIBA instruction is executed. The lower 2 bits of the interrupt address for RIP interrupts are zeros; thus, the interrupt addresses are four locations apart. Three locations following each interrupt address are normally used for the start of an interrupt processing routine.

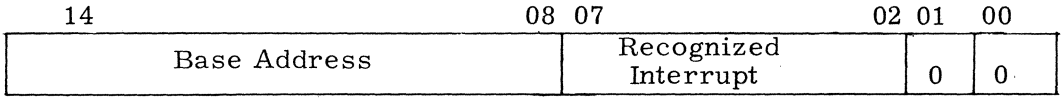


Figure 4-1. Interrupt Address Format

Interrupt addresses correspond to interrupt lines as follows:

Interrupt Line	Interrupt Address
0	XXX XXX X00 000 000
1	XXX XXX X00 000 100
2	XXX XXX X00 001 000
.	.
.	.
.	.
.	.
75	XXX XXX X11 110 100
76	XXX XXX X11 111 000
77	XXX XXX X11 111 100

XXX XXX X = Base Address

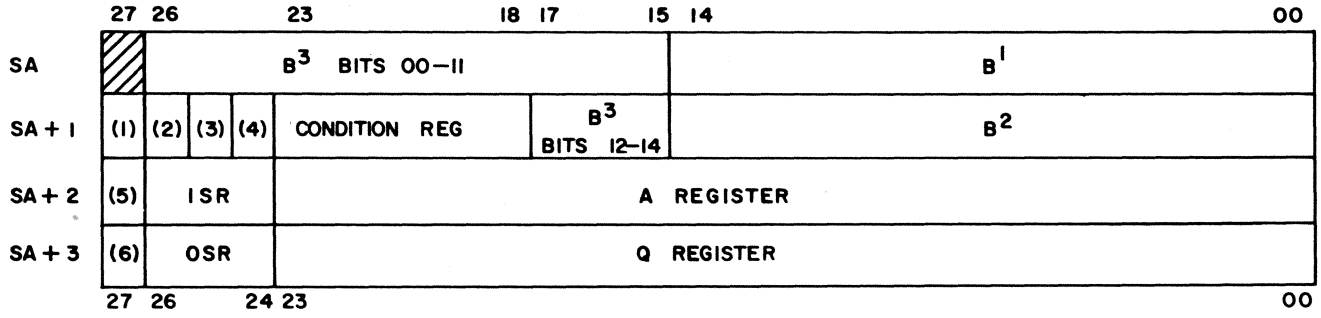
Automatic Register Storage

The RIP contains a 64-word memory, the RIP Register File, which stores the contents of the CPU operational registers. When an interrupt is received on lines $00\text{-}17_8$, the CPU automatically sends the contents of the operational registers to the RIP where they are stored in four register file locations. This shortens interrupt processing time by eliminating the need for several instructions to store operational registers at the beginning of an interrupt routine. Automatic register storage is not provided for lines 20_8 through 77_8.

Four register file locations are associated with each interrupt line as follows:

Interrupt Line	Register File Locations
0	0-3
1	4-7
.	.
.	.
.	.
16	70-73
17	74-77
20	...No...
.	..File ..
.	.. For ..
.	..These.
77	..Lines.

The CPU operational registers stored and their physical placement in the four File locations are shown in Figure 4-2.



(1) BCD Fault
(2) Divide Fault
(3) Arithmetic Overflow Fault
(4) Exponent Overflow/Underflow Fault
(5) Parity bit for word #3
(6) Parity bit for all four words

Figure 4-2. Operational Registers Stored in RIP Register File

In Figure 4-2, 'SA' is the starting address. Upon interrupting, the starting address is determined by the interrupt line recognized.

The operational registers stored in the RIP Register File are not cleared in the CPU with the exception of the Condition register.

Store Operational Registers

The real-time interrupts received by the RIP pass through two stages:

1. The interrupts are sampled and recorded, providing the mask bits for corresponding interrupt lines are set. The recorded lines are then blocked to prevent another sample from affecting them.
2. The RIP recognizes the highest priority interrupt recorded and blocks recognition of any additional interrupts, but continues to record interrupts.

When the RIP recognizes an interrupt, it sends a Real-time Interrupt signal to the CPU. This signal does not require a mask bit in the CPU, nor does it require that the CPU interrupt system be enabled. The RIP also combines the recognized interrupt line and the 7-bit base address to form the interrupt address which is sent to the CPU.

The CPU then accepts the interrupt address from the RIP and stores (P) at this address. If the interrupt line recognized is 00 - 17_8, the storing of registers begins. The RIP receives and stores the CPU operational registers in the appropriate register file locations. Meanwhile, any further interrupt recognition or recording for the recognized interrupt line is prevented, and interrupts for that particular interrupt line are locked out until a Restore instruction is executed. Higher priority interrupts can be recognized provided the RIP is still enabled. When register storage is complete, the CPU begins instruction execution at the interrupt address (from RIP) plus 1.

If the interrupt line recognized is 20_8 - 77_8, the CPU initiates a RNI from the RIP interrupt address +1, and all register storing necessary to regain control following the interrupt routine is under program control.

Restore Operational Registers

At the end of an interrupt routine a restore instruction, Restore (17.00) or Restore and Interrupt (17.04), must be executed. The Restore (17.00) instruction does the following:

1. Activates recognition for the line corresponding to interrupt being processed and all lower priority interrupt lines.
2. Restores the contents of the operational registers, stored in the RIP, to the CPU. This is done only for interrupt lines 00-17_8.

The lower six bits of the instruction must contain the number of the interrupt line in process.

Following the restore operation, the CPU executes the next instruction in sequence. Usually this is a jump instruction used to exit from the interrupt routine. If a jump instruction does not immediately follow the Restore instruction, all real-time interrupts

except RIP Parity Error are disabled until a jump instruction is executed.

The Restore and Interrupt (17.04) instruction performs the same functions as the Restore (17.00) instruction and, in addition, causes a normal interrupt in the CPU providing the Real-Time Interrupt Flag is set. This normal interrupt (Restore interrupt) requires the CPU interrupt system to be enabled before it can be recognized. The Restore interrupt is cleared upon recognition of any normal interrupt including the Restore interrupt.

The Real-Time Interrupt Flag is set upon recognition of a Real-Time Interrupt signal from the RIP and is cleared upon recognition of any normal interrupt including the Restore interrupt.

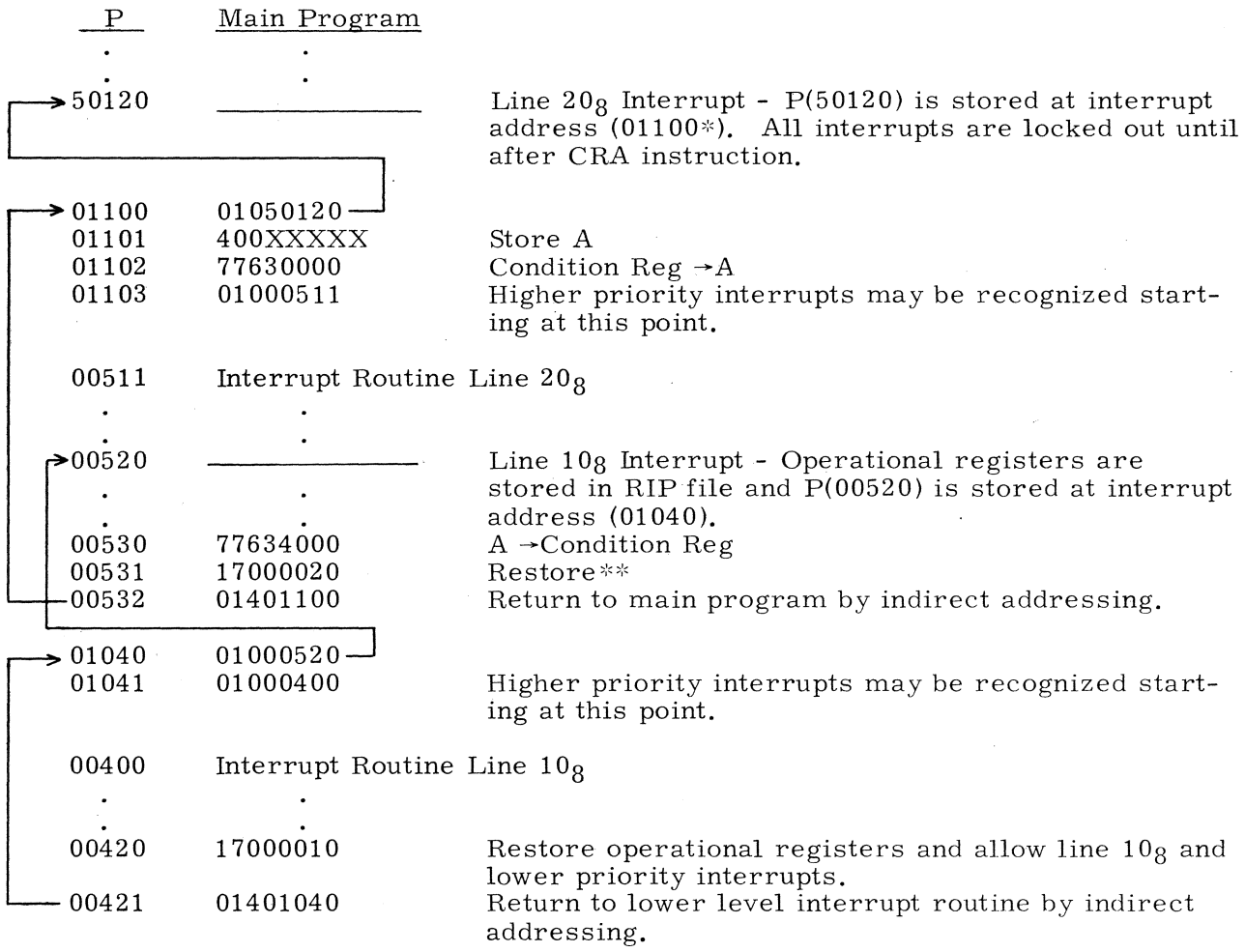
The RSTI instruction informs the operating system that a real-time interrupt has occurred. By causing a normal interrupt in the CPU, the RSTI instruction can prevent a direct return to Program state following the processing of a real-time interrupt.

Condition Register and Priority Considerations

Real-time interrupts on lines 20_8 - 77_8 require the execution of the CRA (Condition Register to A) instruction following the interrupt. However, following interrupts from lines 00 - 17_8, the Condition register is automatically stored in the RIP Register File. Higher priority interrupts can be recognized by the CPU on the first RNI following interrupts on lines 00 - 17_8.

At the end of a RIP interrupt routine, the Condition register must be restored. For interrupt lines 00 - 17_8, a Restore instruction is used to restore the Condition register and other operational registers and disable all interrupts (with the exception of the RIP Parity Error interrupt) until the completion of the next Jump instruction. Following interrupt routines for lines 20_8 - 77_8, both a Restore instruction and an ACR (A to Condition register) instruction are required. The Restore instruction is necessary to allow the current and lower priority RIP interrupts to occur. See the Real-Time Interrupt programming example which follows.

REAL-TIME INTERRUPT PROGRAMMING EXAMPLE



* The upper 7 bits of the interrupt address have been set up previous to the interrupt by executing the (16.02) SIBA instruction. Bits 2 - 7 are determined by the interrupt line. Bits 0 and 1 are always zero.

** This restore does not restore operational registers since none were stored for lines 20_8 - 77_8. It allows recognition of line 20_8 and lower priority interrupts.

RIP Parity Checking

The CPU generates and checks parity on data transmitted between the CPU and the RIP. Parity is generated in the CPU when the operational registers are transmitted to the RIP. Parity is checked when the operational registers are received from the RIP upon execution of a Restore instruction. Each group of four words in the RIP Register File has one parity bit for that group and one parity bit for the third word in that group. A RIP Parity Error can occur only on the instruction immediately following a Restore instruction. A RIP Parity Error causes a Storage Parity Error, one of the CPU abnormal interrupts.

Forced Interrupts

The FRI (16.03) instruction is used to force interrupt conditions in the RIP. The interrupts forced are actual RIP interrupt lines. Thus, all rules which apply to normal RIP interrupts also apply to the forced interrupts.

Program Clear

The CRI (15.0) instruction is a program clear of the RIP and is intended for initialization of the RIP and for diagnostic purposes. It clears the RIP enable, the automatic disable feature, and all recorded and recognized interrupts. It is essentially an internal clear signal from the CPU.

Real-Time Interrupt Response Time

Interrupt response time is defined as the time between the recognition of the real-time interrupt in the RIP and the regaining of control by the CPU at (interrupt address) plus one. During this time, (P) is stored at the interrupt address and for lines 00 - 17_8 the operational registers are stored in the RIP Register File.

The longest instruction time must be included as a part of the response time. The Single Precision Divide is the longest instruction if the floating point instructions are trapped (switch selectable). The response time breakdown, not considering storage conflicts, is as follows:

Case 1. Lines 00-17_8	Divide Execution Time	4.7 μsec
	Store (P)	1.0 μsec
	Store Operational Registers	2.2 μsec
	Interrupt Response Time	7.9 μsec
Case 2. Lines 20_8-77_8	Divide Execution Time	4.7 μsec
	Store (P)	1.0 μsec
	Interrupt Recognition	0.2 μsec
	Interrupt Response Time	5.9 μsec (does not include automatic register storing)

If floating point/double precision instructions are not trapped (FP MODE switch on), the Double Precision Divide instruction (DVAQ) is the longest instruction and must be included in the interrupt response time instead of the Single Precision Divide:

Case 3. Lines $00\text{-}17_8$	DVAQ Execution Time	8.75 μsec
	Store (P)	1.00 μsec
	Store Operational Registers	2.20 μsec
	Interrupt Response Time	11.95 μsec
Case 4. Lines $20_8\text{-}77_8$	DVAQ Execution Time	8.75 μsec
	Store (P)	1.00 μsec
	Interrupt Recognition	0.20 μsec
	Interrupt Response Time	9.95 μsec (does not include automatic register storing)

The storage modules can have up to four access channels. If storage references are being made on the three additional channels during the time in which interrupt response is measured, the interrupt response time is longer. This increase is the time required to service the requests on the other channels. Each additional storage reference requires 0.9 μsec. Cases 1 and 2 could require servicing each of the three additional channels three times, adding 8.1 μsec to the interrupt response time. Cases 3 and 4 could require servicing each of the three additional channels four times, adding 10.8 μsec to the response time.

Interrupt Response time is also affected by block control cycles which may occur while interrupting. Two cycles could occur in cases 1 and 2 and three in cases 3 and 4. One block control cycle can lockout a main control storage request for approximately 1.2 μsec.

INTERRUPT PRIORITY

A definite order of priority exists between the various interrupt conditions in the 3500. The computer begins scanning the priority list from the top as soon as an interrupt signal becomes present. It stops at the first active interrupt encountered (not necessarily the interrupt that initiated the scanning). The computer processes this interrupt and returns to the top of the list where it waits for another active interrupt to again initiate scanning. Table 4-5 lists the order of priority.

TABLE 4-5. INTERRUPT PRIORITY

PRIORITY	TYPE OF INTERRUPT
1	Storage Parity Error
2	Illegal Write
3	Power Failure
4	Real-Time
5	Executive
6	Arithmetic Overflow
7	Divide Fault
8	Exponent Overflow/Underflow
9	BCD Fault
10-73	I/O Equipment (External)*
74-81	I/O Channel**
82	Search/Move
83	Real-Time Clock
84	Manual
85	Associated Processor
86	Restore Interrupt
87	Trapped Instruction

*Each I/O channel monitors eight external equipment interrupt lines; a fully expanded I/O system contains 64 such lines. By representing the 64 lines with the code (L)(Ch), where (L) = line and (Ch) = channel, it is seen that the priority decreases as the code increases. Example: 00 has the highest priority, 01 has second highest, and 77 has the lowest.

**A lower numbered I/O channel interrupt has priority over a higher numbered I/O channel interrupt.

INTERRUPTS DURING EXECUTIVE MODE

Although all interrupts can be recognized during Executive mode, special consideration must be given to handling these interrupts. During Executive mode, the Condition register records current operating information that must be temporarily stored to enable recovery from the interrupt. Table 4-6 lists the Condition register bit assignments.

TABLE 4-6. CONDITION REGISTER BIT ASSIGNMENTS

BIT	CONDITION REPRESENTED
Bit 00	Boundary Jump - Set by SBJP (77.62) instruction Cleared by next jump instruction
Bit 01	Destructive Load A - Set by SDL (77.624) instruction Cleared by next LDA instruction
Bit 02	Operands Relocated Using OSR - Set by ROS (55.4) instruction Cleared by RIS (55.0) instruction
Bit 03	Program State Jump - Set by any jump during Program state Cleared when jumping to Program state.
Bit 04	Interrupt System Enabled - Set by EINT (77.74) instruction Cleared by DINT (77.73) instruction
Bit 05	Program State - Set when jumping to Program state.

To insure the processing of stacked interrupts, it is necessary to transfer these conditions to the A register at the start of the interrupt routine by executing a CRA (77.63) instruction. At the completion of the interrupt routine, these conditions must be restored by executing an ACR (77.634) instruction.

The Condition register is automatically transferred to the RIP Register File upon interrupting on real-time interrupt lines $00\text{-}17_8$. The Condition register is cleared upon completion of the transfer and the CRA instruction is not required. A Restore instruction must be used in place of the ACR instruction for lines $00\text{-}17_8$. This instruction restores the Condition register and other operational registers and disables all interrupts except the RIP Parity Error interrupt until the next Jump instruction is executed.

Upon interrupt recognition, the interrupt system is automatically disabled and the Central Processor enters the Monitor state. The Condition register and most interrupts* are disabled during the intervals between interrupt recognition and CRA instruction execution, and between execution of the ACR instruction and the jump instruction normally used to exit from an interrupt routine.

The Condition register is cleared as the transfer to A is completed; the interrupt system remains disabled until an EINT (77.74) or ACR instruction is executed.

STATUS

Internal/External Status

Internal and external status conditions become available to the programmer by sensing status lines in the system. Internal status reflects conditions within a channel, or within the Central Processor. The INS and INTS instructions are used to sense internal status. Certain internal conditions that can be selected as interrupts also provide status on internal status lines. These internal conditions are cleared upon sensing. External status is sensed with the EXS instruction; the status available on the 12 status lines between the channel and controller varies with the peripheral equipment.

* Exceptions: Trapped instruction interrupts and Real-Time interrupts on lines $00\text{-}17_8$.

If a condition is sensed that is active, a RNI is performed at P+1; if not active a RNI is performed at P+2. Refer to the instruction descriptions for the conditions that can be sensed and the corresponding mask bits.

Illegal Write Status

The Illegal Write condition is an internal condition that provides status. During Non-Executive mode, any attempt to write into the Non-Executive Auto Load/Auto Dump area protected by the storage protect switches causes a "1" to be placed on Internal Status Line 5.

During Executive mode, the status on line 5 is active only if an Illegal Write interrupt occurred during a RNI or RADR sequence.

5. RELOCATION SYSTEM

GENERAL INFORMATION

Every instruction that references Magnetic Core Storage specifies either a word or character address, depending on the operation being performed. A word address consists of 15 bits and uniquely identifies a 24-bit word within a 32K address range. Two additional bits are required during character addressing to select one of four characters within a storage word; however, these bits are not sent to storage.

In Executive mode, the 3500 employs an address relocation technique whereby all addressing is redirected through a hardware index file. A fully expanded storage system of 262,144 words can be accessed by modifying and augmenting the basic 15-bit word address. In Non-Executive mode, relocation does not occur and storage access is limited to 32,768 words. Certain CPU models do not contain the address relocation feature.

This section describes the relocation process and the program protection that is available. Specific information on word and character addressing, address conversion, and forms of address modification are described in Section 6 prior to the instruction list.

RELOCATION PROCESS

Relocation is the process of modifying basic program addresses so that a written program can be located and executed in one of many different storage areas. In the 3500 system, relocation is accomplished through indexing a page-organized storage system. Three bits (contents of ISR or OSR except for I/O and search/move operations where the upper three bits of operand address originate in bits 00 through 02 of the A register) are appended to the basic word address to accomplish addressing of an index file. Constants stored in this file are used in modifying program addresses so that a program can be executed from any available page of memory, regardless of the initial addresses assigned.

Figure 5-1 is a diagram showing the flow of the relocation process. The various elements involved are discussed on the following pages.

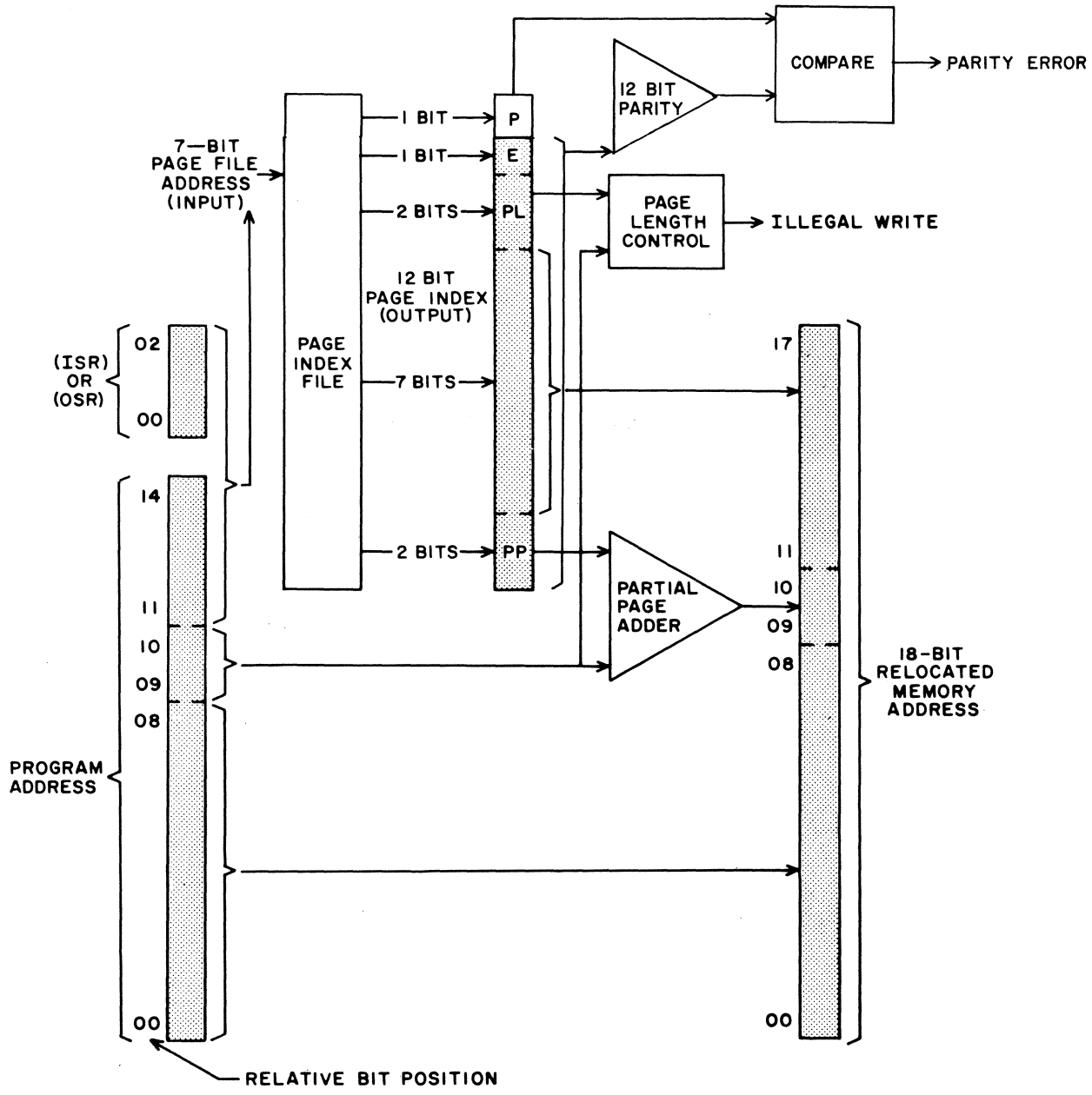


Figure 5-1. Relocation Process

Page Structure

Each page of storage consists of 2,048 absolute storage locations, with a fully expanded system containing 128 such pages. Individual pages are subdivided into quarter (partial) pages of 512 storage locations each. A stored program may occupy more than one full page, or quarter multiples of a page. Addressing in this paging scheme is accomplished through a hardware matrix of FF's called a Page Index File.

Page Index File

The relocation section contains an optional rapid-access flip-flop matrix consisting of 128 14-bit words and is called the Page Index File. This file is referenced to obtain physical page addressing information and to provide memory protection. Only 12 bits of each word (page index) are significant in programming.

During memory operations, the upper 7 bits of the original 18-bit address select the page index to be referenced. Of these 7 bits, the upper 3 bits consist of the contents of the Instruction State or Operand State register. For each setting (0-7) in the ISR or OSR, there is a related block of 16 indexes within the file that is referenced. These eight different groups of 16 indexes are referred to as 'states'.

If the optional Page Index File is not present, the original 18-bit address is sent through the relocation section directly to storage without being altered.

During Monitor state one can read from, or write into any page index (except Page Index Zero*) within the file. This is accomplished by executing either a 77.65 (Page File to A) or 77.64 (A to Page File) instruction. The page file address consists of the lower 7 bits of the instruction word.

Page Index

Figure 5-2 shows the basic format of a 12-bit page index. The significance of each designator is described below.

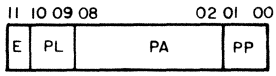


Figure 5-2. Page Index Format

Page Address (PA): Since a fully expanded memory system is organized into 128 pages (177_8) with each page having a unique page address, 7 bits of an index are required to accomplish physical page addressing. The page address becomes the upper 7 bits of the relocated address (bits 11-17).

Partial Page (PP): Each page in memory is subdivided into quarter (partial) pages numbered 0-3. The PP portion of the index (bits 00 and 01) specifies which quarter page is assigned as the starting point for addressing. The assigned quarter page designator is added to bits 09 and 10 of the original address to obtain the actual quarter page where referencing occurs (refer to Partial Page Adder in this section). Figure 5-3 shows the significance of PP bits in selecting starting quarters.

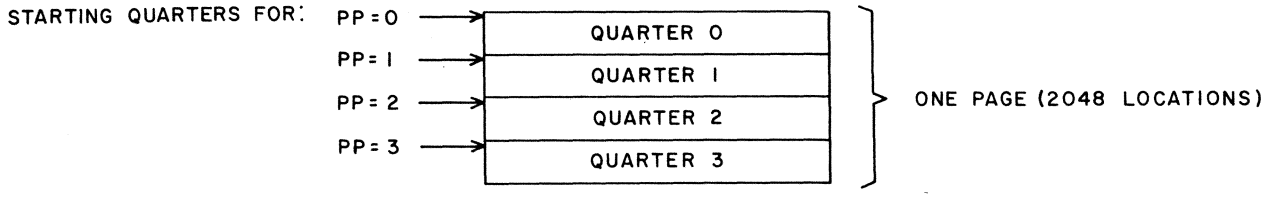


Figure 5-3. Selecting Starting Quarters

* Refer to Page Index Zero Considerations in this section.

Page Length (PL): The PL designator (bits 09 and 10 of the index) defines how many quarter pages of the page at PA (beginning with the quarter specified by PP) can be referenced. Although one page index specifies at most a full page, other indexes may be referenced, or the same one changed, to increase the area allotted a program. Definite boundaries necessary for program or area protection are also established.

The significance of the four designator values is illustrated in Figure 5-4.

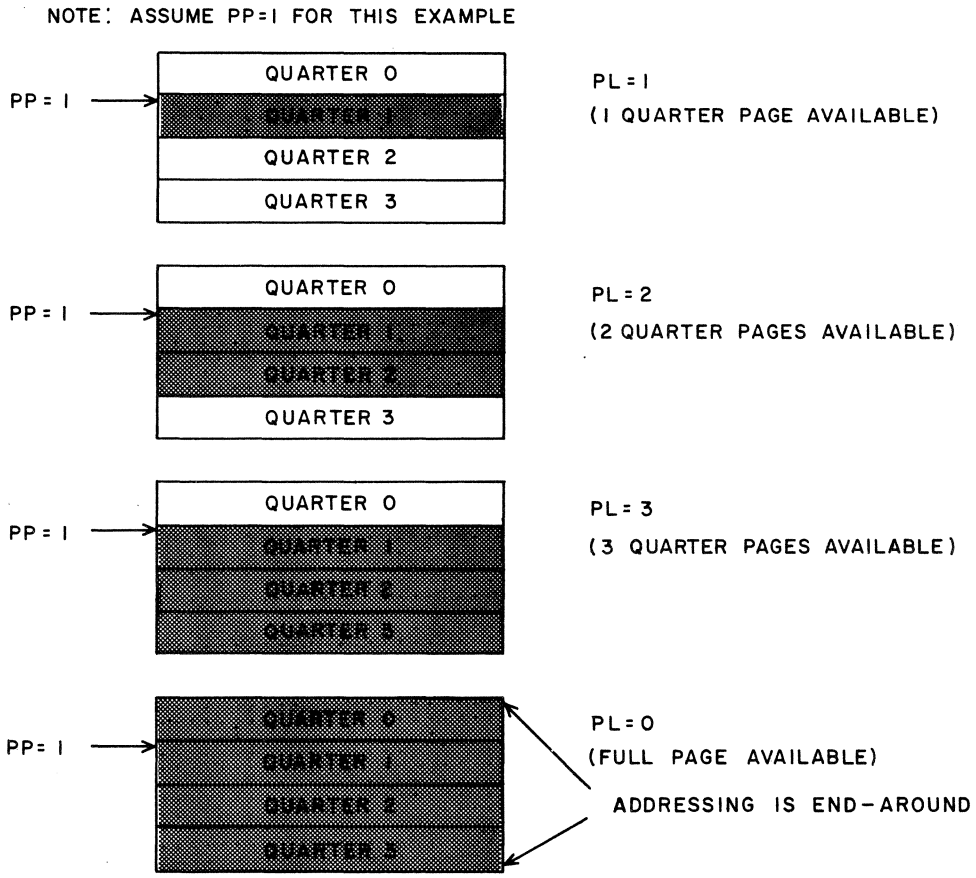


Figure 5-4. Page-Length Designator

Exclusion Bit (E): The Exclusion bit (bit 11) in an index is used as a flag to indicate a protected area within memory. Its interpretation is dependent upon values for addressing designators PA, PL, and PP. One of the three following meanings apply:

1. If E = 0, the values for PA, PL, and PP define a page or partial page where reading or writing is permitted.
2. If E = 1, and the value for PA, PL, or PP is other than zero, a non-destructable page or partial page is defined. Only reading is permitted.
3. If E = 1, and the values for PA, PL, and PP are equal to zero, an unaddressable page is defined. Neither reading nor writing is permitted.

PARTIAL PAGE ADDER

A special adder (2-bit adder with no end-around-carry) is used to combine the PP bits from the page index and bits 09 and 10 of the original address. The partial sum becomes bit 09 and 10 of the relocated address and indicates the quarter in which actual referencing occurs. Example A and Figure 5-5 show the actual quarter in which referencing occurs for specific PL, PP, and bit 09 and 10 values.

NOTE
Refer to discussion on Program Protection later in this section for restrictions on values for PL and bits 9 and 10.

EXAMPLE A: PL = 3
PP = 2
Bits 9 and 10 = 2
Partial Sum = 0

Analysis: Three quarter pages are available. The assigned starting point for addressing is Quarter 2; however actual referencing occurs in Quarter 0.

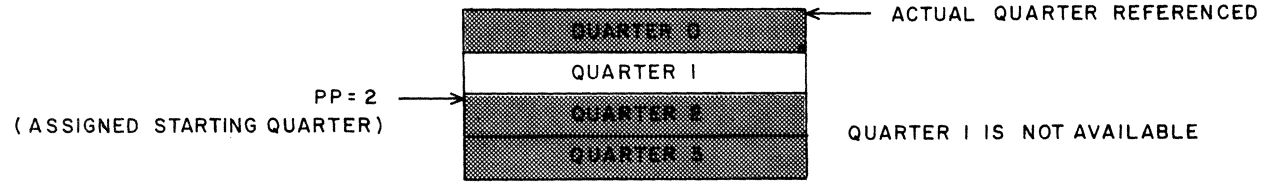


Figure 5-5. Actual Quarter Referenced

PAGE INDEX ZERO CONSIDERATIONS

Page Index Zero is the only index that has a distinct hardware relationship with memory. The PA and PP portions of this index are always zero, whether referenced during Program or Monitor state. Page Index Zero therefore always addresses Page 0 (first physical page) of memory, facilitating storing of the Executive Auto Load/Auto Dump subroutines. The E designator and PL portion of the index may, however, be changed by the monitor.

PROGRAM ADDRESS GROUPS

The lower 15 bits of program addresses define absolute addresses ranging from 00000 through 77777_8. Any program or group of programs within this range of addresses that can be compiled and loaded without conflicting addresses can be considered part of a program address group. The Monitor Program changes the ISR and OSR contents as a program becomes active, thus directing execution of the program through a particular state of the Page Index File. The ISR and OSR define the specific program address group for all memory references except I/O and Search/Move operand references. The 16 indexes within the state are loaded with the proper quantities to perform physical page addressing and to provide program protection. Table 5-1 lists the application of the ISR and OSR registers in reference to the different operating conditions within the Central Processor. For initial Monitor state, and upon any interrupt-return to Monitor state, the upper 3 bits of address are forced to zero to allow return to the monitor program (State 0); however, the ISR and OSR are not altered. Figure 5-6 illustrates program address groups within the paging system.

TABLE 5-1. INSTRUCTION AND OPERAND REFERENCING

OPERATIONAL STATE OF THE PROCESSOR	INSTRUCTIONS REFERENCED WITH	OPERANDS** REFERENCED WITH
Initial Monitor state	Zero	Zero
Monitor state and 55.4 (relocate to operand state) instruction executed	Zero	Contents of OSR
Transition from Monitor state to Program state	Contents of ISR	Contents of ISR*
Program state and 55.4 (relocate to operand state) instruction executed	Contents of ISR	Contents of OSR
Program state and 55.0 (relocate to instruction state) instruction executed	Contents of ISR	Contents of ISR
Any interrupt condition to Monitor state	Zero	Zero

* Transition from Monitor state to Program state does not change the operand address mode.

** EXCEPTIONS:

1. I/O and Search/Move instructions - Upper 3 bits of operand address originate in bits 0-2 of A register regardless of operating mode.
2. 77.75 (Set Typewriter Input) and 77.76 (Set Typewriter Output) instructions - Upper 3 bits of operand address must be preset in Register File 23 (bits 21-23).

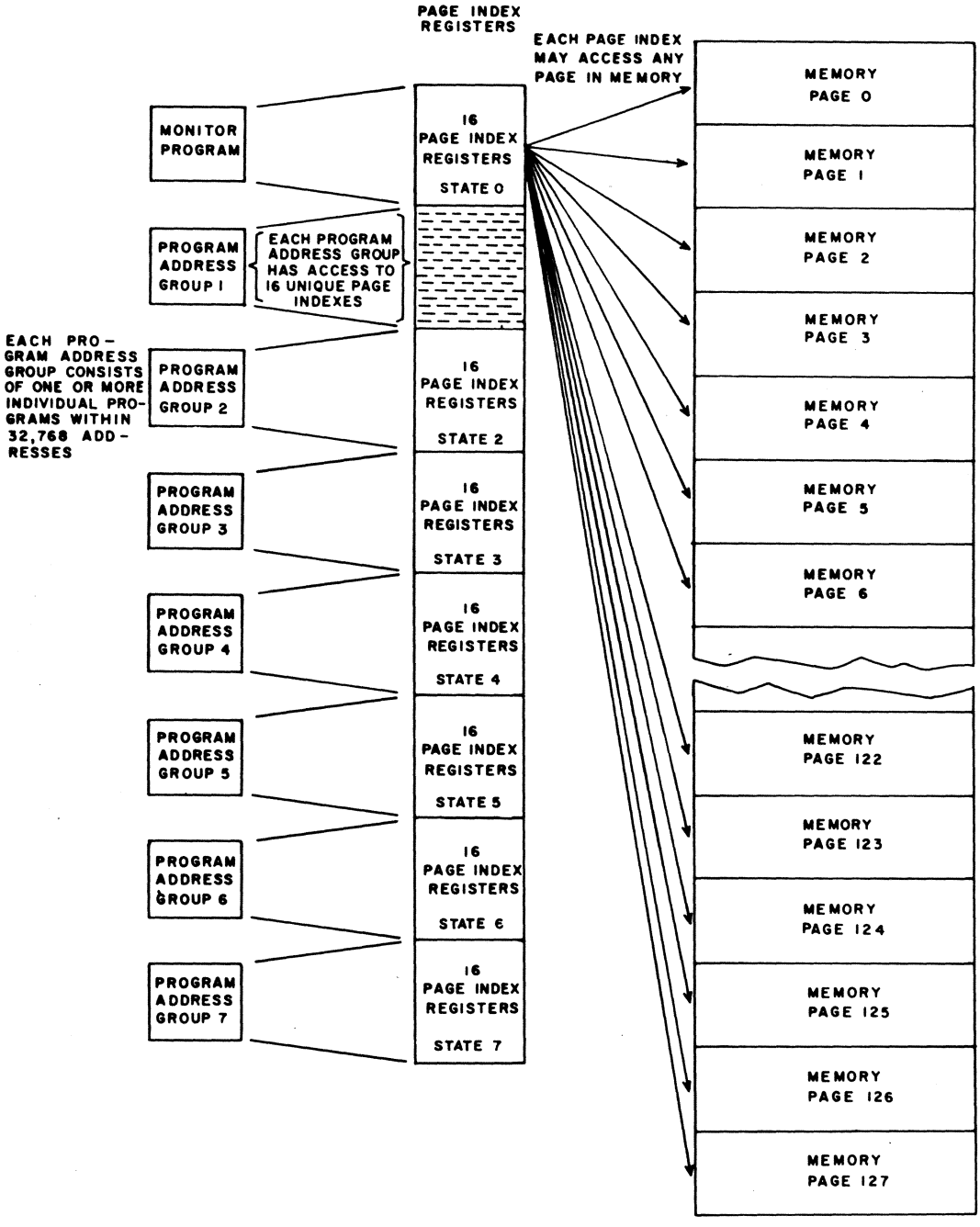


Figure 5-6. Program Address Groups

PROGRAM PROTECTION

The relocation hardware protects against illegal address references by using the Exclusion bit and page length control to assign boundary conditions for a program. As each page index is referenced, tests are made for each of the following illegal conditions:

1. Bits 9 and 10 of the original address specify a quarter page equal to or greater than PL, when PL $\neq$ 0.

   EXAMPLE: Illegal Write would occur if bits 9 and 10 = 3 and PL = 1.
   Illegal Write would not occur if bits 9 and 10 = 3 and PL = 0.

2. A write into storage if the E designator for the referenced index equals 1. Reading is still permitted.

3. A Read or Write with storage if the E designator for the referenced index equals 1 and PA, PL, and PP equal 0. The page is unaddressable.

4. During the first operand reference for a double precision instruction if the second operand reference will satisfy either 1, 2 or 3, as listed above.

Refer to Section 4 for information on when the above conditions cause an Illegal Write interrupt and set the Illegal Write status bit.

A parity bit is generated and written for each 12-bit word written into the Page File. Parity is checked any time the file is referenced. Detection of a file parity error may stop the CPU and/or generate a coded interrupt to the same address as memory parity errors. Main core storage is protected against write references, to wrong addresses, caused by a file failure detected as a parity error.

6. INSTRUCTIONS

GENERAL INFORMATION

A 3500 machine coded instruction word is 24 bits in length. Although most instructions can be coded with a single instruction word, some instructions require up to three words to express execution parameters. This section provides the programmer with a complete hardware description of the instruction set through parameter descriptions, format and addressing requirements, and individual instruction descriptions and examples.

INSTRUCTION PARAMETERS

Following are the parameters used in the 3500 instruction list. In some cases the definitions listed are general in nature and apply to all instructions. Refer to the parameter descriptions given in each instruction description for specific parameter applications.

A = (1) variable length field of characters designated field A in BDP instructions; usually the transmitting field.

(2) in the descriptions for instructions (other than those for BDP) to indicate the A register.

a = addressing mode (a = "0" for direct addressing, a = "1" for indirect addressing)

B = indicates backward storage operation during I/O operations

b = index register designator 1, 2, or 3.

B_m = index register flag for a field in certain BDP instructions. The flag indicates which index register will have its contents added to the unmodified address 'm'. $M = m + [B_m]$ for these instructions only.

B_n = index register flag for a field in certain BDP instructions where both fields are specified by word addresses. The flag indicates which index register will have its contents added to the unmodified address 'n'.

B_r = index register flag for field A in most BDP instructions. Initial character address of field A is defined: $R = r + [B_r]$. If $B_r = 1$ or 3, use (B^1); if $B_r = 2$, use (B^2); if $B_r = 0$, no indexing is performed and $R = r$.

B_s = index register flag for field C in most BDP instructions. Initial character address of field C is defined: $S = s + [B_s]$. If $B_s = 1$ or 3 use (B^1); if $B_s = 2$, use (B^2); if $B_s = 0$, no indexing is performed and $S = s$.

C = variable length field of characters designated field C in BDP instructions. Usually the receiving field.

ch = denotes I/O channel (0-7).

cm = channel mask for channel(s) 0-7 (CILO and CLCA instruction only).

d = interrupt base address designator for RIP instructions.

dc = indicates delimiting character position in instruction word or mnemonic. Generally, a delimiting character of 6 or 8 bits is specified in an instruction and if a character is recognized during the particular operation that equals the delimiting character, the operation is terminated.

g = denotes interrupt group (0-7) for RIP instructions.

G = "1" for word count control with the INPC and INPW instructions.

H = indicates special Assembly/Disassembly operation in certain character oriented I/O instructions.

INT = "1" for interrupt upon completion in certain I/O and Search/Move instructions.

I = assembly language designator indicating indirect addressing.

i = internal parameter (decrement or increment).

j = Jump designator

k = (1) unmodified shift count for SHA, SHQ, and SHAQ instructions

(2) scale factor for SCAQ instruction

K = (1) modified shift count, $K = k + (B^b)$ for SHA, SHQ, and SHAQ instructions.

(2) residue quantity for SCAQ instruction.

l = interrupt line code for RIP instructions.

$ = field length of data block for MOVE instruction.

$1 = number of characters in BDP field A (character count).

$2 = number of characters in BDP field C (character count).

m = unmodified 15-bit storage word address.

M = modified 15-bit storage address. $M = m + (B^b)$.

n = same as 'm', but the word address of the second operand for certain I/O and BDP instructions.

N = indicates special Assembly/Disassembly operation in certain word oriented I/O instructions.

r	=	unmodified 17-bit character address
R	=	modified 17-bit character address: R = r + [B_r] for BDP instructions. (Refer also to 'B_r') R = r + (B^b) for all other instructions
s	=	same as 'r', but the character address of the second operand for certain I/O and BDP instructions.
S	=	(1) modified 17-bit character address of field C for BDP instructions only. S = s + [B_s] (Refer also to 'B_s') (2) mnemonic designator for sign extension in certain instructions.
sc	=	6-bit comparison character used in search instructions. May be used with 'dc'
v	=	a specific register number (00-77) within the Register File
w	=	7-bit Page Index File address. (Refer to APF and PFA instructions and Section 5 for additional information.)
x	=	connect code or interrupt mask
y	=	15-bit operand
z	=	17-bit operand
////	=	slashing indicates a particular area of an instruction that should be loaded with zeros although the particular area is not used for the instruction.

In addition to the instruction parameters, abbreviations are used in the instruction descriptions to refer to various registers and operations. These abbreviations and their literal meanings are listed here:

(A_{00-02})	=	Contents of the lower 3 bits (00, 01, and 02) of the A register
BCR	=	BDP Condition register
[B_r]	=	contents of the index register as determined by the value of the flag B_r
CIR	=	Channel Index register
CR	=	Condition register
ISR	=	Instruction State register
m < n	=	m "less than" n
m > n	=	m "greater than" n
$m \geq n$	=	m "greater than or equal to" n
$m \wedge n$	=	logical product of m and n
$(m) \rightarrow n$	=	contents of m, transferred to n
RADR	=	Read Address (indirect address) sequence
RNI	=	Read Next Instruction word sequence; one, two, or three such sequences may be required to obtain the complete instruction from memory.

ROP	=	Read Operand sequence
STO	=	Store Operand sequence
V	=	Exclusive OR function
Λ	=	AND functions

ADDRESSING AND INSTRUCTION FORMATS

Word Addressing

Figure 6-1 illustrates the bit assignments for a word-addressed instruction. Reference the instruction parameter for description of the symbols used. Instructions using word addressing usually allocate the lower 15 bits of the instruction word for an unmodified storage word address. Usually a word address may be modified before instruction execution by adding to it the contents of a designated index register (called indexing). When 'm' represents a 15-bit word address, the modified address 'M' then equals 'm' + (B^b). A different type of modification (indirect addressing) is sometimes available, in which the lower 18 bits of the instruction word are replaced by the lower 18 bits of memory location 'm' prior to execution. These two modification processes are discussed and illustrated later in this section.

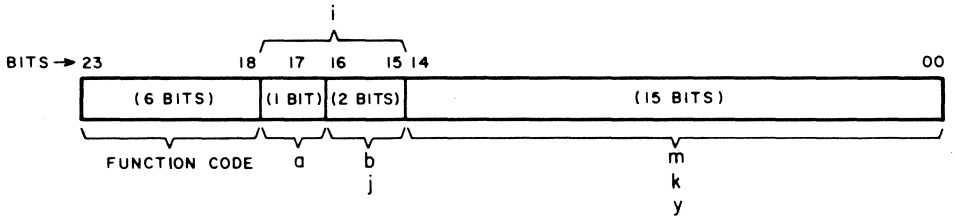
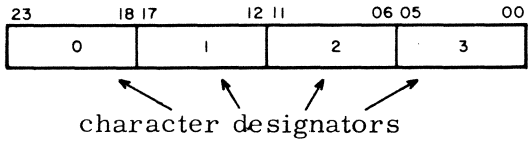


Figure 6-1. Word-Addressed Instruction Format

Other instructions with similar formats use the lower 15 bits to specify an operand or shift count.

Character Addressing

Seventeen bits from an instruction word (bits 00-16) are used to address 6-bit characters within storage. This address consists of a 15-bit word address (bits 02-16) and a 2-bit character designator. These lower 2 bits designate the character in the storage word as follows:



The Central Processor generates the necessary control signals for manipulation of character-size data. During Executive mode operation, 3 address bits are appended to enable access to every character within the storage system.

Indirect addressing cannot be used with character addressed instructions, however indexing is possible in some instances (see LACH instruction). Usually when indexing is available, a predetermined index register must be used.

Figure 6-2 illustrates the format of an instruction using character addressing. Other instructions with similar formats use the lower 17 bits as an operand. Reference the instruction parameters for definition of the symbols used.

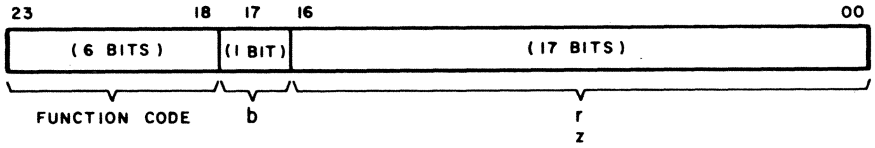


Figure 6-2. Character-Addressed Instruction Format

The optional business-oriented instructions (64-67 codes) require three instruction words to completely define an operand. These instruction words are always located in consecutive memory locations (P, P+1, and P+2). The majority of BDP instructions use character addressing. The lower 17 bits of P and/or P+1 are used. If word addressing is used, bits 02-16 specify the word address (refer to Direct Address information later in this section). Figure 6-3 illustrates the format for most BDP instructions. Refer to Instruction Parameters for symbol definitions and to the individual BDP instructions for existing format differences.

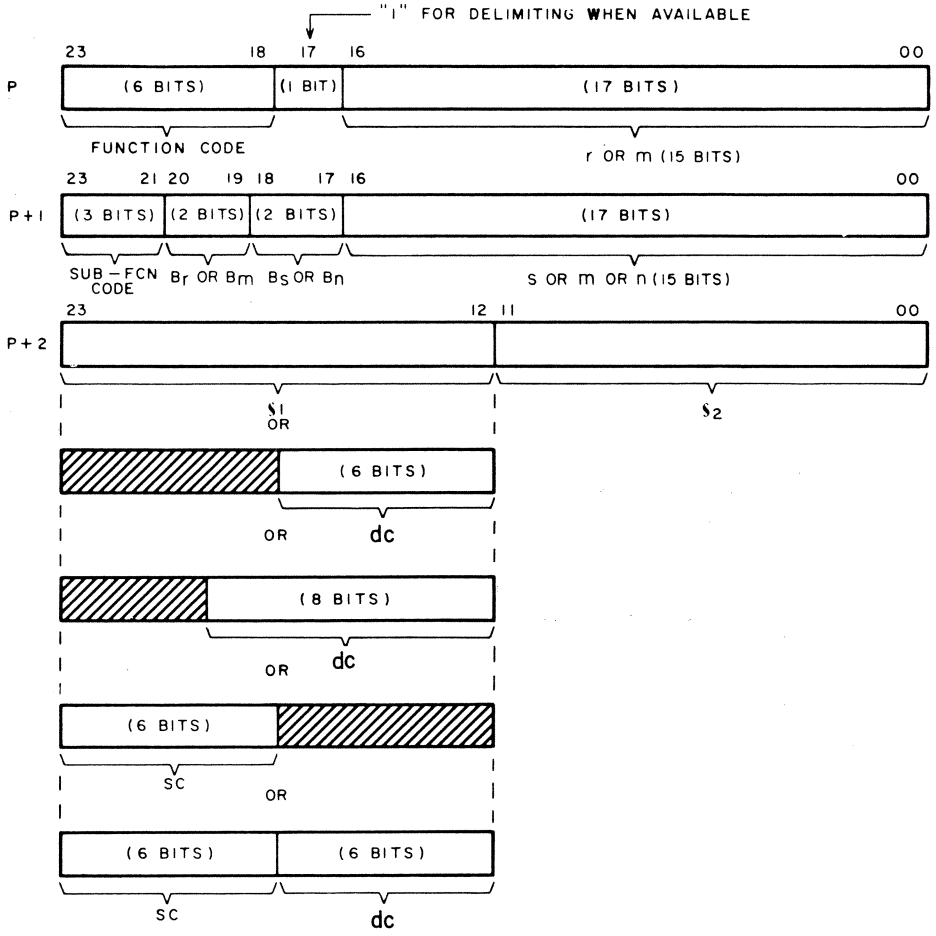


Figure 6-3. BDP Instruction Format

RIP Instruction Format

Figure 6-4 illustrates the bit assignments for the optional Real-Time Interrupt Processor instructions. Refer to Instruction Parameters for symbol definitions and to the individual RIP instructions for existing format differences.

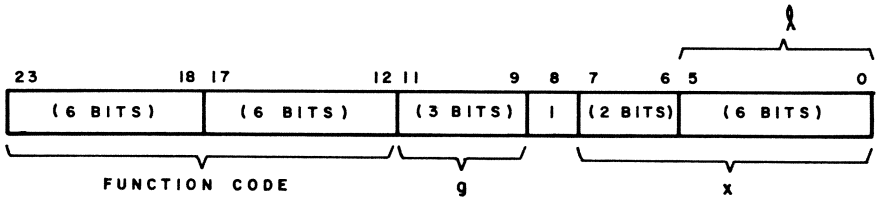


Figure 6-4. RIP Instruction Format

Address Conversion

It is often desirable to convert a word address and character position to its corresponding character address or vice versa. The following procedure is a technique used for this purpose.

To convert a word address to a character address:

- Octally multiply the word address by four. (During program execution, this operation is simulated by a left shift of two binary places.)
- Add the character position to the product.

The sum is the character address.

EXAMPLE: Given: Word address 12442, character position 2
Find: Corresponding character address

1) 12442
x4
52210

2) +2
52212 = character address

To convert a character address to a word address:

- Octally divide the character address by four

The quotient is the word address and the remainder is the character position. No remainder indicates character zero.

EXAMPLE: Given: Character address 03442
Find: Word address and character position

```
   00710 = word address
4 ) 03442
    34
    4
     4
     2 = remainder = character position 2
```

Addressing Modes

Three modes of addressing are used in the computer: No Address, Direct Address, and Indirect Address.

No Address

This mode is used when an operand 'y' or a shift count 'k' is placed directly into the lower portion of an instruction word. Symbols 'a' and 'b' are not used as addressing mode and index designators with any of the no address instructions.

Direct Address

The direct addressing mode is used in any instruction in which an operand or instruction address 'm' is stored in the lower portion of the initial instruction word. This mode is specified by setting bit 17 of the instruction ('a' designator) equal to 0.

During direct addressing, the contents of an index register may be added (indexed) to a word or character address to obtain a modified execution address. Generally, when word addressing is used, a 2-bit designator 'b' (bits 15 and 16 of the instruction) specify which of the three index registers is to be used. For character addressing, a predetermined index register is declared when bit 17 of the instruction equals 1.

NOTE
Special indexing considerations apply to certain BDP instructions as described below.

An index register flag (B_r, B_s, B_m, or B_n) is used to indicate which index register is used to modify the A or C field address.

IF: $s = 00413$, $B_s = 2$, and $(B^2) = 00364$.

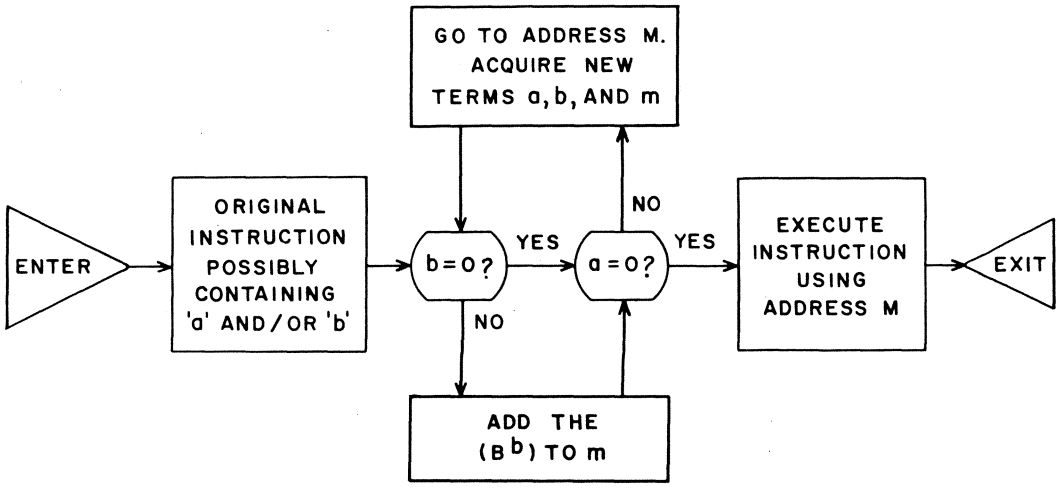
THEN: $S = s + [B_s]$
$S = 00413 + (B^2)$
$S = 00413 + 00364 = 00777$

Certain BDP conversion instructions (PAK, UPAK, DTA) use both word and character addressing. The lower 2 bits of the instruction specifying word addressing are not a part of the address, but are still added to the contents of the specified index register when doing indexing. They must be set to zero to prevent a normal arithmetic carry during addition. The lower 2 bits of the index register should be set to 1's to allow for end-around-carry.

Indirect Address

Indirect addressing is used only with those instructions that have an execution address 'm'. For applicable instructions, indirect addressing is specified by setting bit 17 ('a' designator) equal to 1. Several levels of indirect addressing (intermixed with indexing) may be used to reach an execution address; however, execution time is delayed in direct proportion to the number of levels used. The search for a final address continues until 'a' equals 0. The sequence of steps taken by the hardware to perform an indirect address is called a RADR (Read Address) sequence.

It is important to note that indirect addressing and indexing are two distinct and independent steps, one of which may be specified without the other. The figure below shows the basic sequence followed and priority of modification. Examples of indirect addressing and indexing are given on the following page.



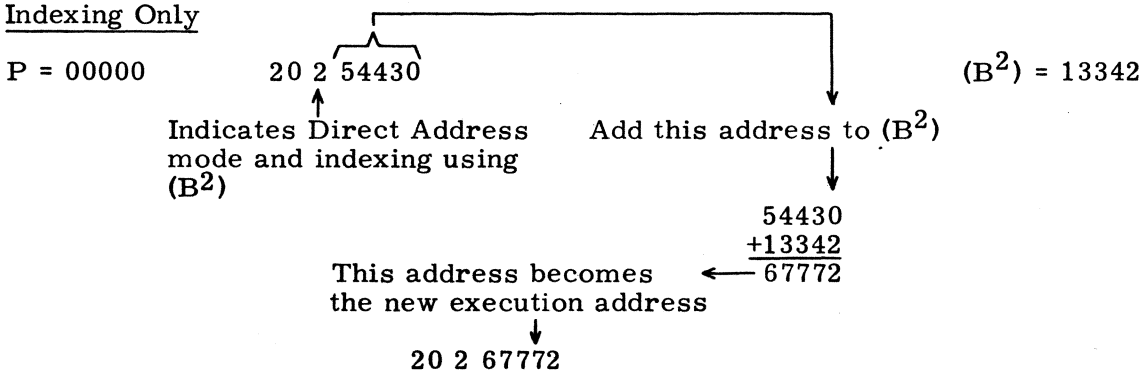
NOTE

Unless it is otherwise stated, indirect addressing follows the above routine throughout the list of instructions.

Indexing and Indirect Addressing Examples

The following examples use the LDA (20) instruction; however, the process applies to any of the instructions with an 'a' and/or 'b' designator.

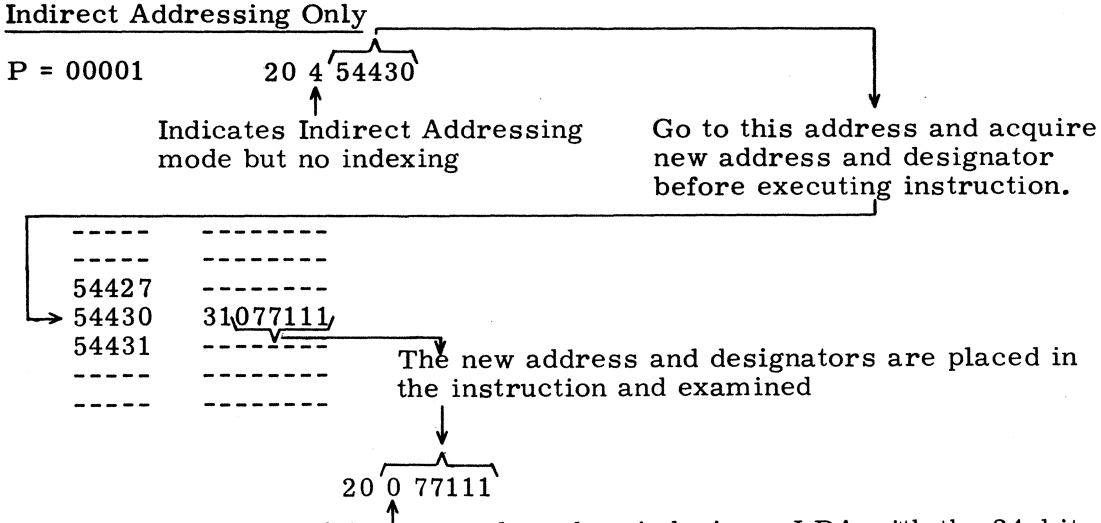
Indexing Only



LDA with the 24-bit quantity stored at address 67772

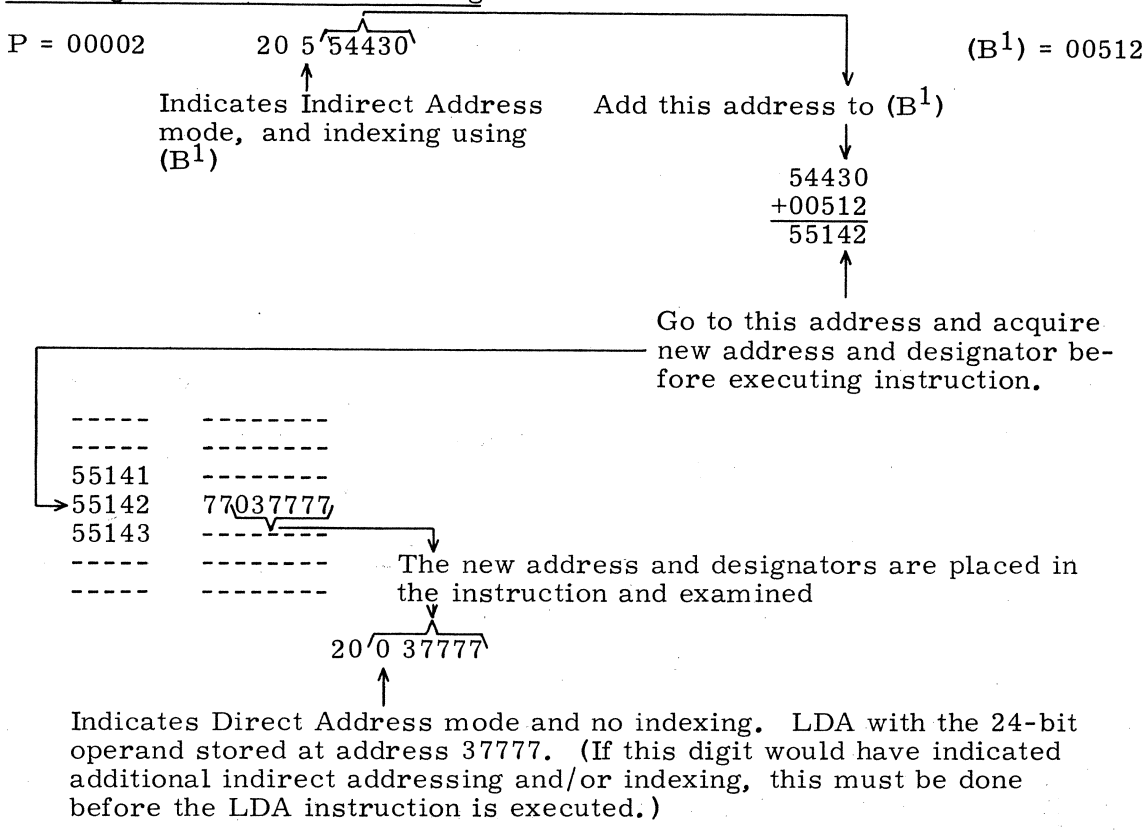
```
    67771    --------
P = 67772    77700000 ──→ This quantity is loaded into the A register
    67773    --------
```

Indirect Addressing Only



Indicates Direct Address mode and no indexing. LDA with the 24-bit operand stored at address 77111. (If this digit would have indicated additional indirect addressing and/or indexing, this must be done before the LDA instruction is executed.)

Indexing and Indirect Addressing



INSTRUCTION LIST

The instructions are grouped according to their basic function (Loads, Stores, Arithmetic) with the exception of PROGRAM AND RELOCATION CONTROL and BUSINESS DATA PROCESSING instructions, which form their own distinct groups. Cross-referencing is provided from group to group where necessary.

Each group of instructions is introduced with an index, and if necessary, a group description. This description contains pertinent information regarding characteristics and programming restrictions of the instruction group.

Individual instructions are presented according to the following format:

- Heading, which includes the assembly language mnemonic and instruction name or function
- Machine code instruction format
- Parameters and their descriptions
- Instruction description
- Comments (when necessary)

Complete octal and mnemonic instruction listings are available in the last section of this manual. A quick reference index of the different instruction groups is located inside the back cover.

No-Operation Instructions

When an attempt is made to execute one of the following instructions at the current execution address (P) the computer recognizes them as No-Operation (No-Op) instructions and advances to the next execution address (P+1). Other machine language codes representing No-Op instructions are not listed. In COMPASS mnemonics, a No-Op instruction is written as NOP.

NO-OPERATION
OCTAL CODES

02.0
14.0

If the computer is operating in Non-Executive mode, the instructions listed below are recognized as No-Operation instructions.

NON-EXECUTIVE NO-OPERATION INSTRUCTIONS

ACI	CRI	JAA	SAD
ACR	CRIM	OSA	SBJP
AIS	DAD	PFA	SDL
AOS	DRIP	RIS	SIBA
APF	ERIP	ROS	SRIM
CIA	FRI	RST	TMAV
CRA	ISA	RSTI	

Trapped Instructions

The optional BDP instructions are trapped if the BDP MODE switch is OFF. Double precision multiplication/division and all Floating Point instructions are trapped if the FP MODE switch is OFF. Both the BDP MODE and the FP MODE switches are located on the console. Refer to Trapped Instruction Interrupts and Table 4-4 in Section 4 for additional information.

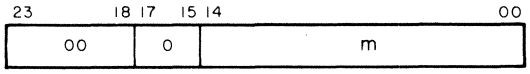
Halt and Stop Instructions

Operation Field		Address Field	Interpretation
HLT	00.0	m	Halt
SLS	77.70		Selective stop
UCS	77.77		Unconditional stop

NOTE

An Executive interrupt is generated if an attempt is made to execute any of these instructions during Program State of Executive mode (reference Section 4).

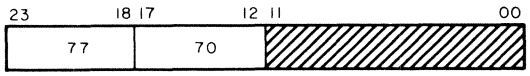
HLT Halt



Instruction Description: Unconditionally halt at this instruction. Upon restarting, RNI from address 'm'.

Comments: Indirect addressing and indexing may not be used.

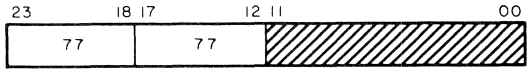
SLS Selective Stop



Instruction Description: Program execution halts if the SELECT STOP switch on the console is set. RNI from address P + 1 upon restarting.

Comments: Bits 00 through 11 should be loaded with zeros.

UCS Unconditional Stop



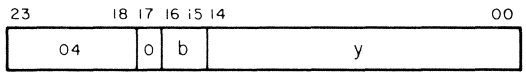
Instruction Description: This instruction unconditionally stops the execution of the current program. RNI from address P + 1 upon restarting.

Comments: Bits 00 through 11 should be loaded with zeros.

Skip Instructions

Operation Field		Address Field	Interpretation
ISE	04.0-3	y, b	Skip next instruction if (B^b) = y
ASE	04.6	y	Skip next instruction if (A) = y
ASE, S	04.4	y	Skip next instruction if (A) = y. Sign of y is extended.
QSE	04.7	y	Skip next instruction if (Q) = y
QSE, S	04.5	y	Skip next instruction if (Q) = y. Sign of y is extended.
ISG	05.0-3	y, b	Skip next instruction if (B^b) $\geq$ y
ASG	05.6	y	Skip next instruction if (A) $\geq$ y
ASG, S	05.4	y	Skip next instruction if (A) $\geq$ y. Sign of y is extended.
QSG	05.7	y	Skip next instruction if (Q) $\geq$ y
QSG, S	05.5	y	Skip next instruction if (Q) $\geq$ y. Sign of y is extended.
ISI	10.1-3	y, b	Index skip, incremental
ISD	10.4-7	y, b	Index skip, decremental

ISE
Skip Next Instruction if (B^b) = y

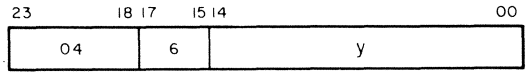


b = index register designator

Instruction Description: If (B^b) ='y', skip to address P+2; if not, RNI from address P+1.

Comments: If b = 0, 'y' is compared to zero.

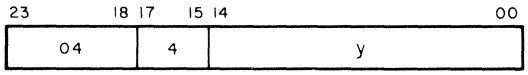
ASE
Skip Next Instruction if (A) = y



Instruction Description: If (A) = 'y', skip to address P+2; if not, RNI from address P+1.

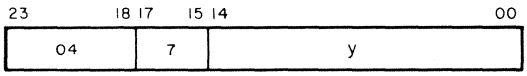
Comments: Only the lower 15 bits of A are used for this instruction.

ASE, S
Skip Next Instruction if (A) = y, Sign Extended



Instruction Description: Same as ASE except the sign of 'y' is extended. All 24 bits of A are recognized.

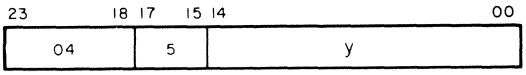
QSE
Skip Next Instruction if (Q) = y



Instruction Description: If (Q) ='y,' skip to address P+2; if not, RNI from address P+1.

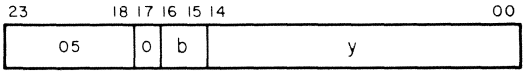
Comments: Only the lower 15 bits of Q are used for this instruction.

QSE, S
Skip Next Instruction if (Q) = y, Sign Extended



Instruction Description: Same as QSE except the sign of 'y' is extended. All 24 bits of Q are recognized.

ISC
Skip Next Instruction if (B^b) ≧ y

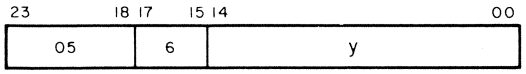


b = index register designator

Instruction Description: If (B^b) are equal to or greater than 'y', skip to address P+2; if not, RNI from address P+1. (B^b) and 'y' are 15-bit positive numbers.

Comments: If b = 0, 'y' is compared to zero.

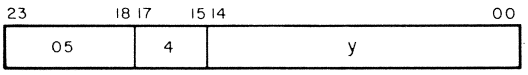
ASG
Skip Next Instruction if (A) ≧ y



Instruction Description: If (A) are equal to or greater than 'y', skip to address P+2; if not, RNI from address P+1. Only the lower 15 bits of A are used.

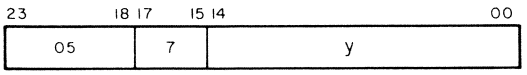
Comments: (A_{L15}) and 'y' are considered 15-bit positive numbers.

ASG, S
Skip Next Instruction if (A) ≧ y, Sign Extended



Instruction Description: Same as ASG except the sign of 'y' is extended. All 24 bits of A are recognized. Positive zero (00000000) is recognized as greater than negative zero (77777777).

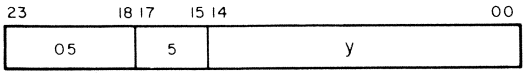
QSG
Skip Next Instruction if (Q) ≧ y



Instruction Description: If (Q) are equal to or greater than 'y', skip to address P+2; if not, RNI from address P+1. Only the lower 15 bits of Q are used.

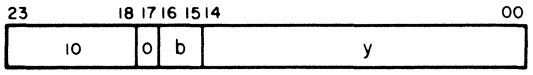
Comments: (Q_{L15}) and 'y' are considered 15-bit positive numbers.

QSG, S
Skip Next Instruction If (Q) ≧ y, Sign Extended



Instruction Description: Same as QSG except the sign of 'y' is extended. All 24 bits of Q are recognized. Positive zero (00000000) is recognized as greater than negative zero (77777777).

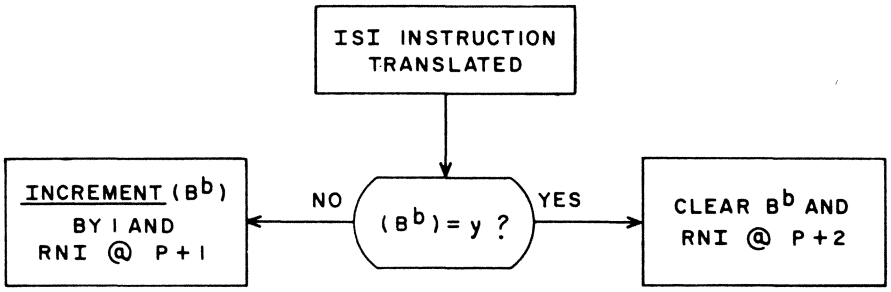
ISI
Index Skip, Incremental



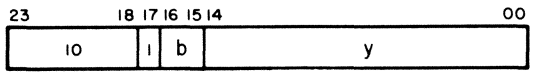
b = index register designator

Instruction Description: If $(B^b) = {}'y'$, clear B^b and skip to address P+2; if not, add one to (B^b) and RNI from address P+1.

Comments: The 10.0 instruction is a SSH (storage shift) instruction, described later in this section. Positive zero (00000) and negative zero (77777) form an equal comparison.



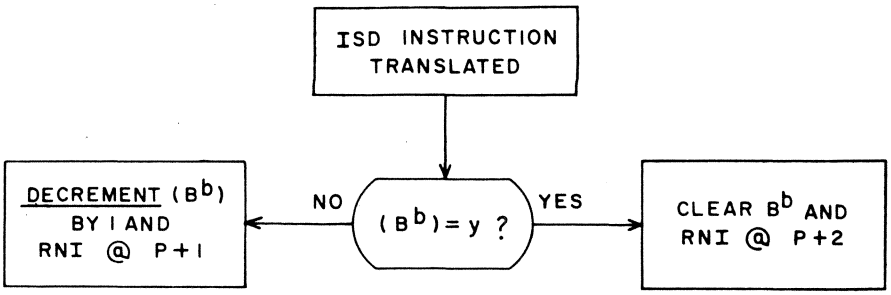
ISD
Index Skip, Decremental



b = index register designator

Instruction Description: If $(B^b) = {}'y'$, clear B^b and skip to address P+2; if not, subtract one from (B^b) and RNI from address P+1.

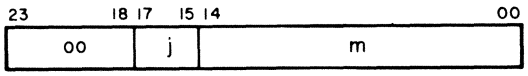
Comments: Positive zero (00000) and negative zero (77777) form an equal comparison.



Jump Instructions

Operation Field		Address Field	Interpretation
SJ1-6	00.1-6	m	Jump if appropriate key (1-6) is set
RTJ	00.7	m	Return jump
UJP, I	01	m, b	Unconditional jump
IJI	02.0-3	m, b	Index jump; Incremental index
IJD	02.4-7	m, b	Index jump; Decremental index
AZJ	03.0-3	m	Compare A with zero for Jump condition
AQJ	03.4-7	m	Compare A with Q for Jump condition
JMP, HI	70.0	m	Jump on Positive result
JMP, ZRO	70.1	m	Jump on Zero result
JMP, LOW	70.2	m	Jump on Negative result

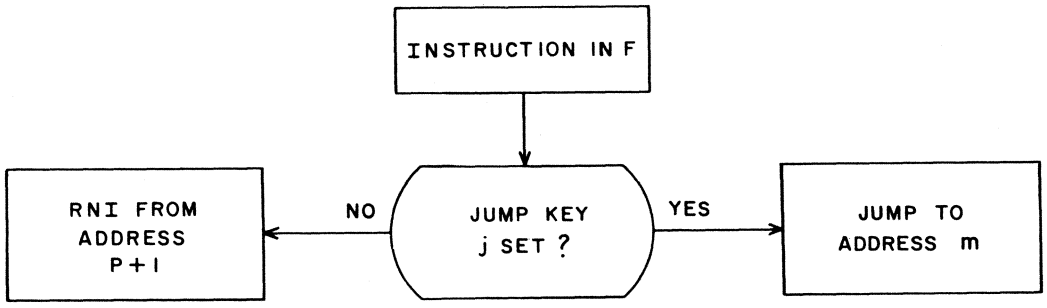
SJ 1-6
Selective Jump



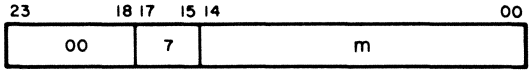
j = jump keys 1 to 6
m = jump address

Instruction Description: Jump to address 'm' if Jump key 'j' is set; otherwise, RNI from address P+1.

Comments: Indirect addressing and indexing may not be used.



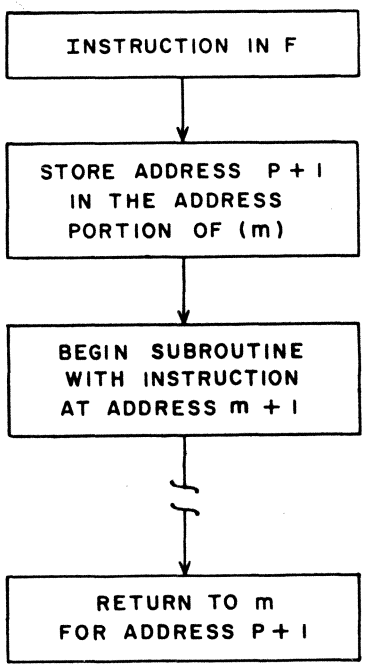
RTJ
Return Jump

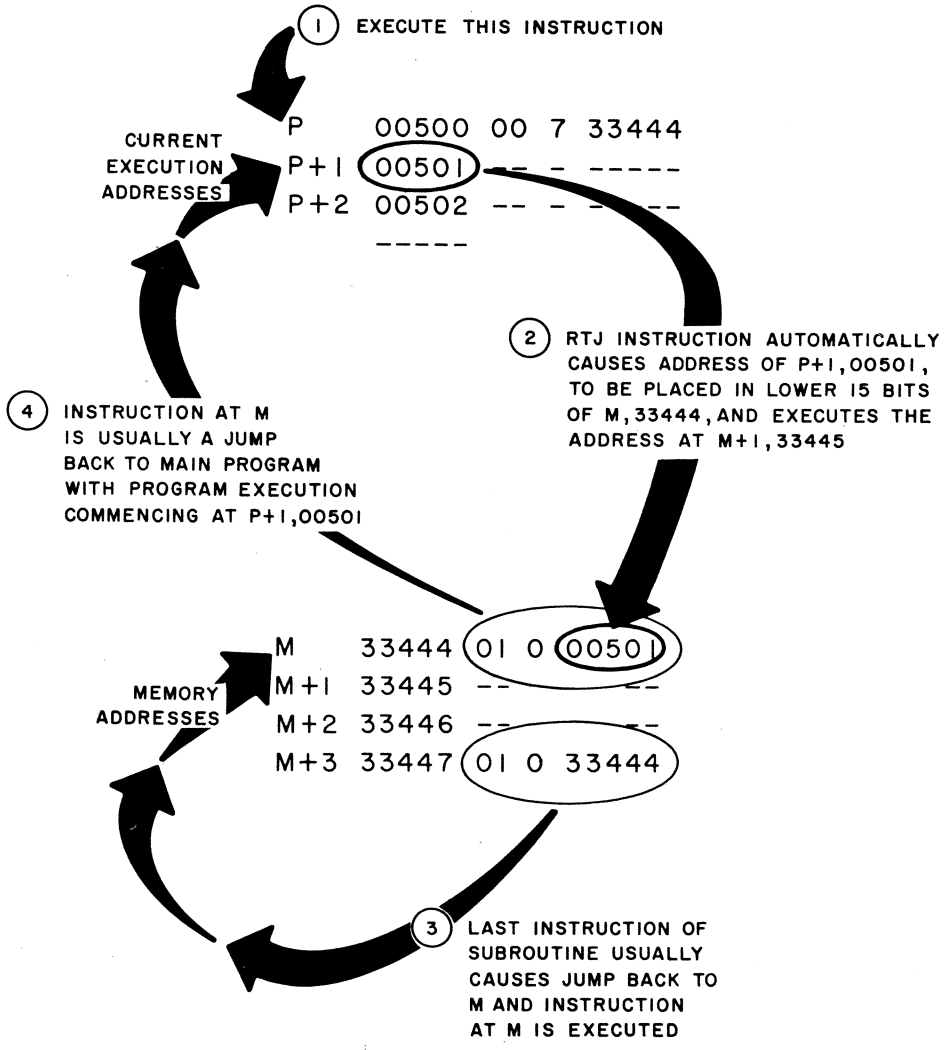


m = jump address

Instruction Description: The address portion of location m is replaced with the return address P+1. Jump to location 'm+1' and begin executing instructions at that location.

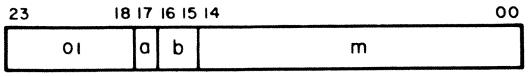
Comments: The (ISR) are used for addressing location 'm' during the STO cycle. This instruction should not be used to transfer control from Monitor state to Program state. If a RTJ instruction is executed and the Boundary Jump flag is set (refer to SBJP instruction), the STO cycle is executed in Monitor state, that is, address P+1 is stored at address 'm' of the monitor program. Indirect addressing and jump address indexing may not be used. An example of an executed RTJ instruction is illustrated on the following page.





RTJ EXECUTION EXAMPLE

UJP
Unconditional Jump

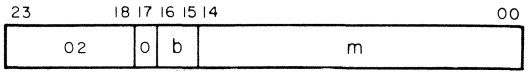


a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Unconditionally jump to address M.

Comments: Indirect addressing and indexing may be used.

IJI
Index Jump, Incremental

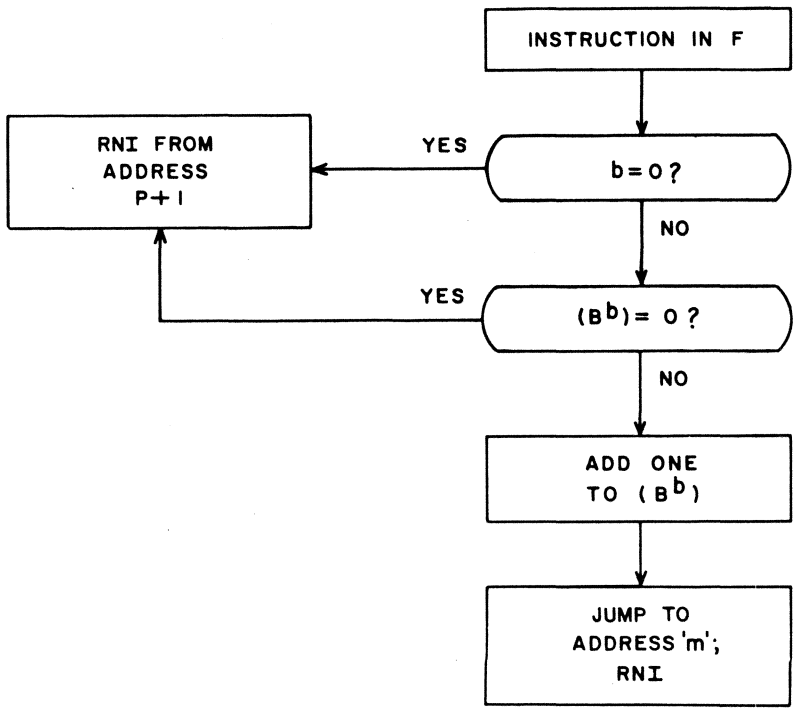


b = index register designator
m = jump address

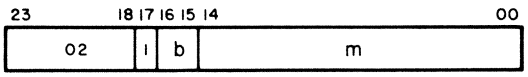
Instruction Description: If b = 1, 2, or 3, the respective Index register is examined:

1. If $(B^b) = 00000$, the jump test condition is not satisfied; RNI from address P+1.
2. If $(B^b) \neq 00000$, the jump test condition is satisfied. One is added to (B^b); jump to address 'm' and RNI.

Comments: If b = '0', this is a No-Operation instruction; RNI from address P+1. Indirect addressing and jump address indexing may not be used. The counting operation is done in a one's complement additive accumulator. Negative zero (77777) is not generated because the count progresses from: 77775, 77776, to 00000 (positive zero) and stops. If negative zero is initially loaded into B^b, the count progresses: 77777, 00001, 00002, etc. In this case, the counter must increment through the entire range of numbers to reach positive zero.



IJD
Index Jump, Decremental

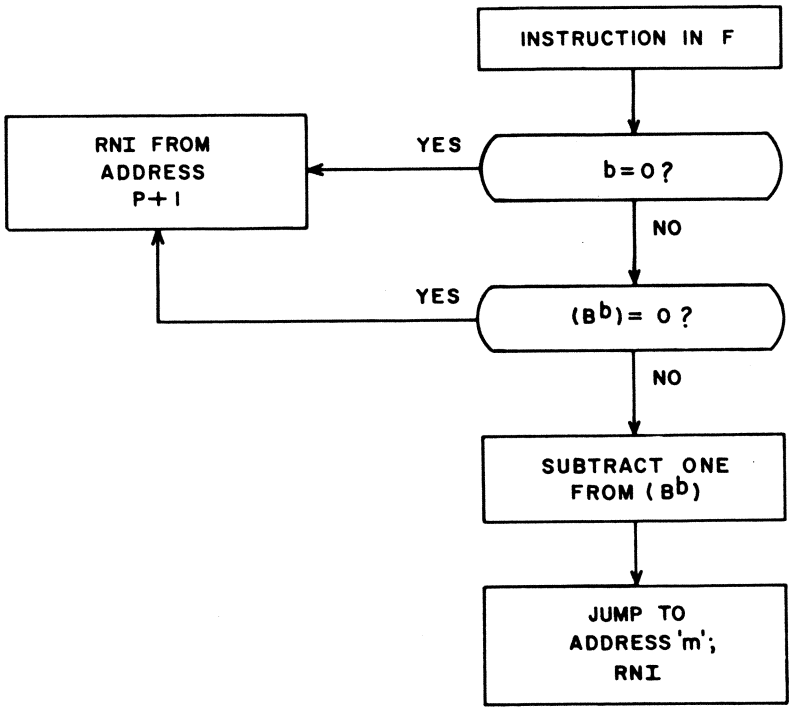


b = index register designator
m = jump address

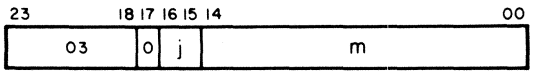
Instruction Description: If b = 1, 2, or 3, the respective Index register is examined:

1. If (B^b) = 00000, the jump test condition is not satisfied; RNI from address P+1.
2. If $(B^b) \neq$ 00000, the jump test condition is satisfied. One is subtracted from (B^b); jump to address 'm' and RNI.

Comments: If b = 0, this is a No-Operation instruction; RNI from address P+1. Indirect addressing and jump address indexing may not be used. If negative zero (77777) is initially loaded into B^b, the count decrements through the entire range of numbers to reach 00000 before the program will RNI from P+1.



AZJ
Condition Compare A with Zero, Jump

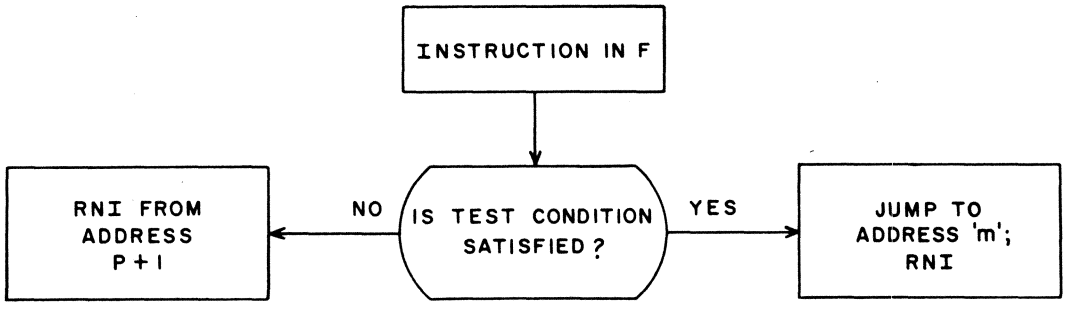


j = jump designator (0-3)
m = jump address

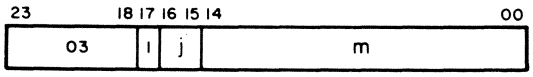
Instruction Description: The operand in A is algebraically compared with zero for an equality, inequality, greater-than or less-than condition (see table). If the test condition is satisfied, program execution jumps to address 'm'. If the test condition is not satisfied, RNI from address P+1.

Comments: Positive zero (00000000) and negative zero (77777777) give identical results when j = 0 or 1. When j = 2 or 3, negative zero is recognized as less than positive zero. Indirect addressing and jump address indexing may not be used.

Condition Mnemonic	Jump Designator j	Test Condition
EQ	0	(A) = 0
NE	1	(A) ≠ 0
GE	2	(A) ≥ 0
LT	3	(A) < 0



AQJ
Condition Compare A with Q, Jump

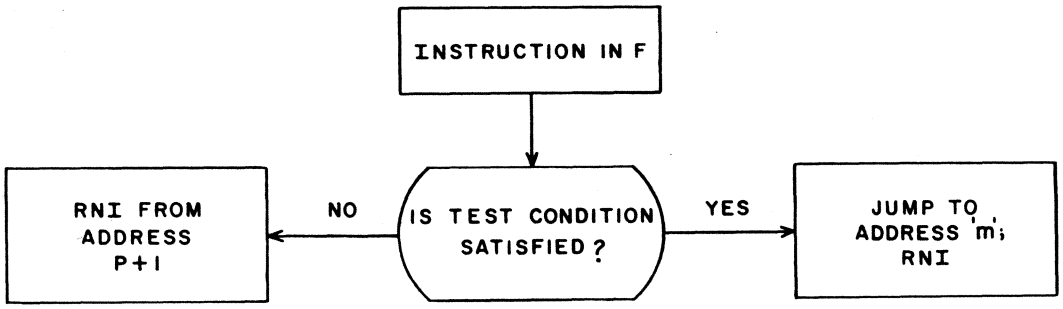


j = 0-3 jump designator (0-3)
m = jump address

Instruction Description: The quantity in A is algebraically compared with the quantity in Q for equality, inequality, greater-than or less-than condition (see table). If the test condition is satisfied, program execution jumps to address 'm'. If the test condition is not satisfied, RNI from address P+1.

Comments: This instruction may be used to test (Q) by placing an arbitrary value in A for the comparison. Positive and negative zero give identical results in this test when j = 0 or 1. When j = 2 or 3, negative zero is recognized as less than positive zero. Indirect addressing and jump address indexing may not be used.

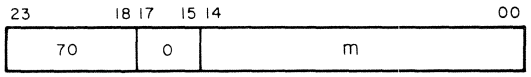
Condition Mnemonic	Jump Designator j	Test Condition
EQ	0	(A) = (Q)
NE	1	(A) ≠ (Q)
GE	2	(A) ≥ (Q)
LT	3	(A) < (Q)



NOTE

The instructions on this page are trapped if the BDP MODE switch on the operator's console is not active.

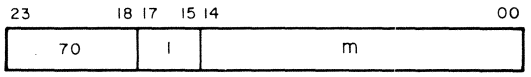
JMP, HI
Jump if result is + or High



m = jump address

Instruction Description: Sense the status of the BDP Condition register (BCR). Jump to address 'm' if the register contains a 01_2 code. Refer to Table 6-1 for conditions represented by the code.

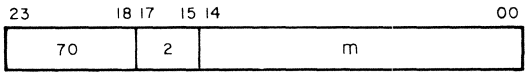
JMP, ZRO
Jump if result = 0



m = jump address

Instruction Description: Sense the status of the BDP Condition register (BCR). Jump to address 'm' if the register contains a 00_2 code. Refer to Table 6-1 for conditions represented by the code.

JMP, LOW
Jump if result – or Low



m = jump address

Instruction Description: Sense the status of the BDP Condition register (BCR). Jump to address 'm' if the register contains a 10_2 code. Refer to Table 6-1 for conditions represented by the code.

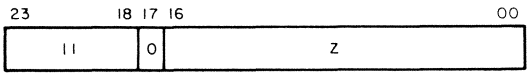
TABLE 6-1. BDP CONDITION REGISTER CODES

BCR Code	Significance After CMP	Significance After ADM & SBM	Significance After TST	Significance After Others
00_2	Fld A=Fld C	Result Positive	Fld A=0	Operand Pos
01_2	Fld A>Fld C		Fld A Pos	
10_2	Fld A<Fld C	Result Negative	Fld A Neg	Operand Neg

Enter and Increase Instructions

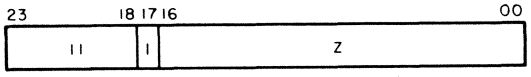
Operation Field		Address Field	Interpretation
ECHA	11.0	z	Enter A with z
ECHA, S	11.1	z	Enter A with z and extend sign of z
ENI	14.0-3	y, b	Enter index b with y
ENA	14.6	y	Enter A with y
ENA, S	14.4	y	Enter A with y and extend sign of y
ENQ	14.7	y	Enter Q with y
ENQ, S	14.5	y	Enter Q with y and extend sign of y
INI	15.0-3	y, b	Increase index by y
INA	15.6	y	Increase A by y
INA, S	15.4	y	Increase A by y, sign extended
INQ	15.7	y	Increase Q by y
INQ, S	15.5	y	Increase Q by y, sign extended

ECHA
Enter Character Address into A



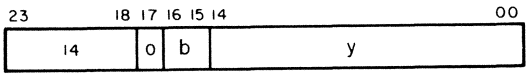
Instruction Description: Clear A; then enter a 17-bit operand 'z' (usually a character address) into A.

ECHA, S
Enter Character Address into A, Sign Extended



Instruction Description: Clear A; then enter a 24-bit operand (17-bit 'z' plus 7 bits of sign extension) into A.

ENI
Enter Index with y

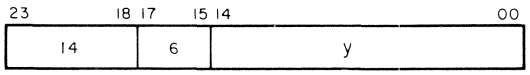


b = index register designator

Instruction Description: Clear index register B^b and enter 'y' directly into B^b.

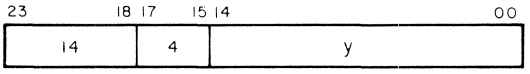
Comments: If b='0', this is a No-Operation instruction.

ENA
Enter A with y



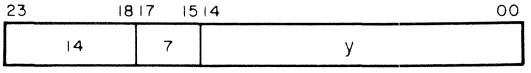
Instruction Description: Clear the A register and enter 'y' directly into A.

ENA, S
Enter A with y, Sign Extended



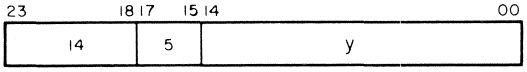
Instruction Description: Same as ENA except the sign of 'y' is extended.

ENQ
Enter Q with y



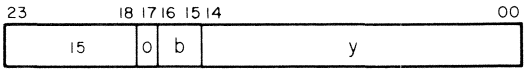
Instruction Description: Clear the Q register and enter 'y' directly into Q.

ENQ, S
Enter Q with y, Sign Extended



Instruction Description: Same as ENQ except the sign of 'y' is extended.

INI
Increase Index by y



b = index register designator

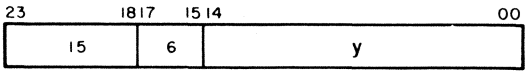
Instruction Description: Add y to (B^b).

Comments: If b = '0', this is a No-Operation instruction. Signs of 'y' and B^b are extended. *If the optional RIP is present, RIP MODE selected, and execution* takes place in Monitor state of Executive mode, b=0 denotes a Clear RIP (CRI) instruction rather than a No-Op. Refer to the Real-Time Interrupt Instructions.

NOTE

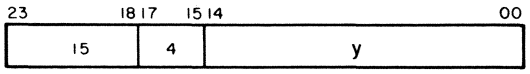
Arithmetic overflow is sensed for these instructions.

INA
Increase A by y



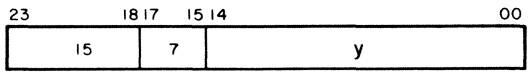
Instruction Description: Add 'y' to (A).

INA, S
Increase A by y, Sign Extended



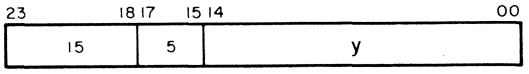
Instruction Description: Same as INA except the sign of 'y' is extended.

INQ
Increase Q by y



Instruction Description: Add 'y' to (Q).

INQ, S
Increase Q by y, Sign Extended



Instruction Description: Same as INQ except the sign of 'y' is extended.

The following examples illustrate the use of sign extension in some instructions:

EXAMPLE A: To enter negative zero into Q, execute a 14 5 77777.

EXAMPLE B: To increase (A) by -17, execute a 15477760 instruction.

```
          (A) = 00066667  (arbitrary value)
                77777760
                --------
                00066647
(end around carry)      1
                --------
                00066650  (A) after instruction execution
```

In all cases of sign extension, bit 14 (for 15-bit 'y' operands) and bit 16 (for 17-bit 'z' operands) determine the sign of the quantity.

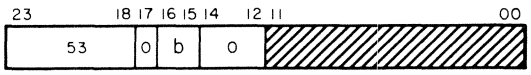
Inter-Register Transfer Instructions

Operation Field		Address Field	Interpretation
TIA	53.(0-3)0	b	Transfer (B^b) to A
TAI	53.(4-7)0	b	Transfer (A) to B^b
TMQ	53.01	v	Transfer (Register v) to Q
TQM	53.41	v	Transfer (Q) to Register v
TMA	53.02	v	Transfer (Register v) to A
TAM	53.42	v	Transfer (A) to Register v
TMI	53.(0-3)3	v, b	Transfer (Register v) to B^b
TIM	53.(4-7)3	v, b	Transfer (B^b) to Register v
AQA	53.04		Transfer (A) + (Q) to A
AIA	53.(0-3)4	b	Transfer (A) + (B^b) to A
IAI	53.(4-7)4	b	Transfer (B^b) + (A) to B^b
ELQ	55.1		Transfer (E_L) to Q
EUA	55.2		Transfer (E_U) to A
EAQ	55.3		Transfer (E_UE_L) to AQ
QEL	55.5		Transfer (Q) to E_L
AEU	55.6		Transfer (A) to E_U
AQE	55.7		Transfer (AQ) to E_UE_L

NOTE

Inter-register transfer instructions with 77 codes are described in the Program and Relocation Control Instruction group.

TIA
(B^b) → A

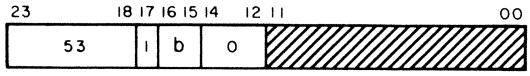


b = index register designator

Instruction Description: Transfer the (B^b) to A.

Comments: Bits 00 through 11 should be loaded with zeros. No sign extension on B^b. Prior to the transfer, (A) are cleared. If b = '0', zeros are transferred to A.

TAI
$(A) \rightarrow B^b$

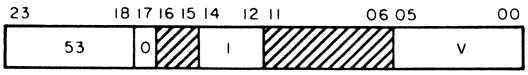


b = index register designator

Instruction Description: Transfer the (A) to B^b.

Comments: Bits 00 through 11 should be loaded with zeros. The (A) remain unchanged. If b = 0, this becomes a No-Operation instruction.

TMQ
$(v) \rightarrow Q$

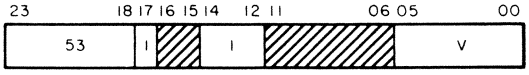


v = register file number, $00\text{-}77_8$

Instruction Description: Transfer the (v) to Q.

Comments: Bits 06 through 11, 15 and 16 should be loaded with zeros.

TQM
$(Q) \rightarrow v$

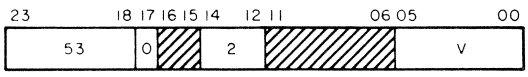


v = register file number, $00\text{-}77_8$

Instruction Description: Transfer the (Q) to 'v'.

Comments: Bits 06 through 11, 15 and 16 should be loaded with zeros. A transfer into one of the file locations $00\text{-}37_8$ causes an Executive interrupt if operating in Program state of Executive mode (reference Section 4).

TMA
$(v) \rightarrow A$

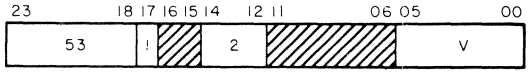


v = register file number, $00\text{-}77_8$

Instruction Description: Transfer the (v) to A.

Comments: Bits 06 through 11, 15 and 16 should be loaded with zeros.

TAM
$(A) \rightarrow v$

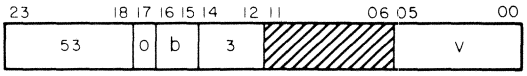


v = register file number, $00\text{-}77_8$

Instruction Description: Transfer the (A) to 'v'.

Comments: Bits 06 through 11, 15 and 16 should be loaded with zeros. A transfer into one of the file locations $00\text{-}37_8$ causes an Executive interrupt if operating in Program state of Executive mode (reference Section 4).

TMI
$(v) \rightarrow B^b$

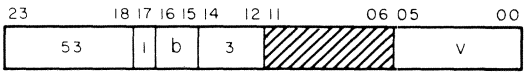


b = index register designator
v = register file number, $00\text{-}77_8$

Instruction Description: Transfer the lower 15 bits of (v) to B^b.

Comments: Bits 06 through 11 should be loaded with zeros. If b = '0', this becomes a No-Operation instruction.

TIM
$(B^b) \rightarrow v$

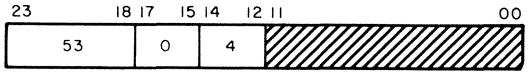


b = index register designator
v = register file number, $00\text{-}77_8$

Instruction Description: Transfer (B^b) to 'v'. The upper 9 bits of 'v' are cleared.

Comments: Bits 06 through 11 should be loaded with zeros. If b = '0', all of (v) are cleared. A transfer into one of the file locations $00\text{-}37_8$ causes an Executive interrupt if operating in Program state of Executive mode (reference Section 4).

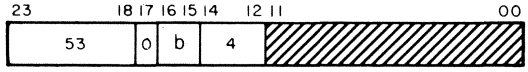
AQA
$(A) + (Q) \rightarrow A$



Instruction Description: Add the (A) to the (Q) and transfer the sum to A.

Comments: (Q) remain unchanged. Bits 00 through 11 should be loaded with zeros. Arithmetic overflow is not sensed.

AIA
$(A) + (B^b) \rightarrow A$

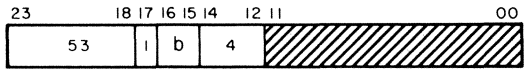


b = index register designator

Instruction Description: Add the (A) to the (B^b) and transfer the sum to A.

Comments: Bits 00 through 11 should be loaded with zeros. The sign of (B^b) is extended prior to the addition. (B^b) remain unchanged. Arithmetic overflow is not sensed.

IAI
$(B^b) + (A) \rightarrow B^b$

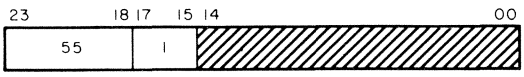


b = index register designator

Instruction Description: Add the (A) to the (B^b) and transfer the sum to B^b.

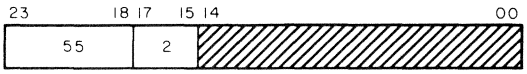
Comments: Bits 00 through 11 should be loaded with zeros. The sign of the original (B^b) is extended prior to the addition. The upper 9 bits of the sum are lost when the sum is transferred to the Index register. Arithmetic overflow is not sensed. If b = '0', this becomes a No-Operation instruction.

ELQ
$(E_L) \rightarrow Q$



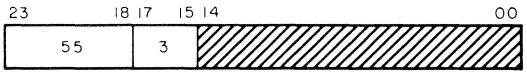
Instruction Description: Transfer the (E_L) to Q. Bits 00-14 should be loaded with zeros.

EUA
$(E_U) \rightarrow A$



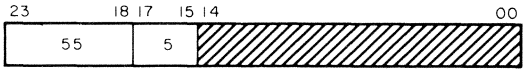
Instruction Description: Transfer the (E_U) to A. Bits 00 - 14 should be loaded with zeros.

EAQ
$(E) \rightarrow AQ$



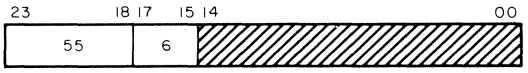
Instruction Description: Transfer the (E_UE_L) to AQ. Bits 00 - 14 should be loaded with zeros.

QEL
$(Q) \rightarrow E_L$



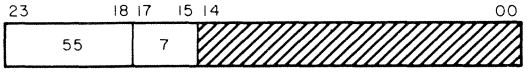
Instruction Description: Transfer the (Q) to E_L. Bits 00 - 14 should be loaded with zeros.

AEU
$(A) \rightarrow E_U$



Instruction Description: Transfer the (A) to E_U. Bits 00 - 14 should be loaded with zeros.

AQE
$(AQ) \rightarrow E$

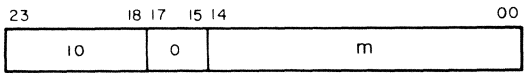


Instruction Description: Transfer the (AQ) to E_UE_L. Bits 00 - 14 should be loaded with zeros.

Shift and Scale Instructions

Operation Field		Address Field	Interpretation
SSH	10.0	m	Storage Shift
SHA	12.0-3	k, b	Shift A
SHQ	12.4-7	k, b	Shift Q
SHAQ	13.0-3	k, b	Shift AQ
SCAQ	13.4-7	k, b	Scale AQ

SSH Storage Shift

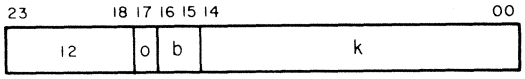


m = storage address

Instruction Description: Sense bit 23 of the quantity stored at address m. If bit 23 = '0' (positive), RNI from P + 1; if negative ("1"), RNI from P + 2. Shift (m) one place left, end-around, and replace it in this same storage location.

Comments: Indexing and indirect addressing may not be used.

SHA Shift A



b = index register designator
k = unmodified shift count; K = k + (B^b)

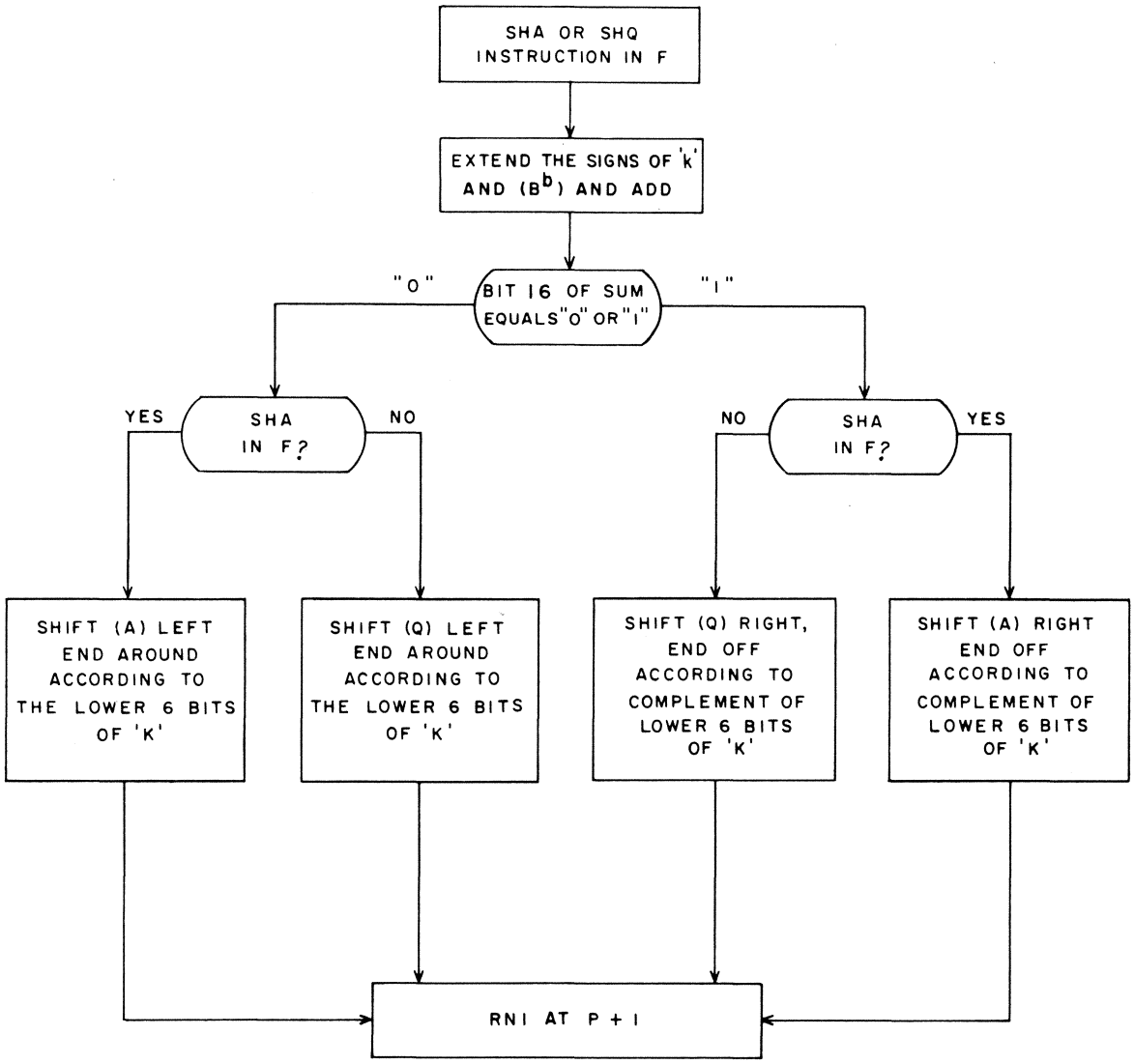
Instruction Description: (B^b) and 'k', with their signs extended, are added. If 'b' = 0, the sign of 'k' is still extended. The sign and magnitude of the 17-bit sum determine the direction and magnitude of the shift. The computer senses only bits 00-05 and 16 of the sum for this information. For left shifts, the shift magnitude is the lower 6 bits of K; for right shifts, the complement of the lower 6 bits of K equals the shift magnitude.

EXAMPLE: Shift left six positions: K = 00000006
Shift right six positions: K = 77777771

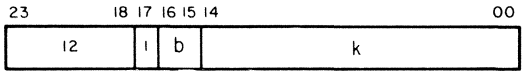
Comments: During left shifts, bits reaching the upper bit position of the A (during SHA) or Q (during SHQ) registers are carried end-around. Therefore, a left shift of 24 places results in no change in (A) or (Q). A left shift that exceeds 24 places results in an effective shift of K-24 places.

During right shifts, the sign bit is extended and the bits are shifted end-off. A right shift of 23 or more places results in (A) or (Q) becoming all "0's" or all "1's", depending upon the original sign.

SHA/SHQ FLOW CHART



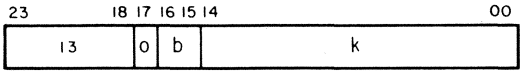
SHQ
Shift Q



b = index register designator
k = shift count: K = k + (B^b)

Instruction Description: Shift (Q). Refer to SHA description.

SHAQ
Shift AQ



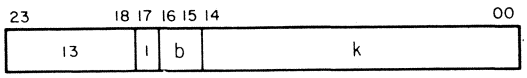
b = index register designator
k = unmodified shift count; K = k + (B^b)

Instruction Description: Shift (AQ). (B^b) and 'k', with their signs extended, are added. If 'b' = 0, the sign of 'k' is still extended. The sign and magnitude of the 17-bit sum determine the direction and magnitude of the shift. The computer senses only bits 00-05 and 16 of the sum for this information. For left shifts, the shift magnitude is the lower 6 bits of K; for right shifts, the complement of the lower 6 bits of K equals the shift magnitude.

EXAMPLE: Shift left three places: K = 00000003
Shift right three places: K = 77777774

Comments: During left shifts, bits reaching the upper bit position of the A register are carried end-around to the lowest bit position of Q. Therefore, a left shift of 48 places results in no change in (AQ). A left shift exceeding 48 places results in an effective shift of K-48 places. During right shifts, the sign bit is extended and the bits are shifted end-off. A right shift of 47 or more places results in (AQ) becoming all "0's" or all "1's", depending upon the original sign.

SCAQ
Scale AQ



b = index register designator
k = scale factor
K = residue = k minus the number of shifts
If b = 1, 2, or 3, then K → B^b

Instruction Description: (AQ) are shifted left, end-around, until the 2 highest order bits are unequal. If (AQ) initially equals positive or negative zero, 48_{10} shifts are executed before the instruction terminates. During scaling, the computer counts the number of shifts. If 'b' = 0, K is discarded; if 'b' = 1, 2, or 3, K is transferred to the designated index register.

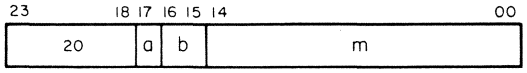
Load Instructions

Operation Field		Address Field	Interpretation
LDA, I	20	m, b	Load A
LDQ, I	21	m, b	Load Q
LACH	22	r, 1	Load A, Character
LQCH	23	r, 2	Load Q, Character
LCA, I	24	m, b	Load A, Complement
LDAQ, I	25	m, b	Load AQ
LCAQ, I	26	m, b	Load AQ, Complement
LDI, I	54	m, b	Load Index

NOTE

The LDL, I instruction is described in the Logical Instructions group later in this section.

LDA
Load A

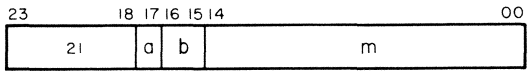


a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Load A with a 24-bit quantity from the storage address specified by M.

Comments: Indirect addressing and indexing may be used. This instruction becomes a destructive load if bit 01 of the Condition register is set (see SDL instruction).

LDQ
Load Q

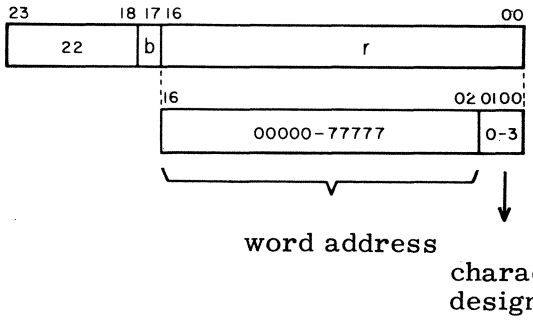


a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Load Q with a 24-bit quantity from storage address M.

Comments: Indirect addressing and indexing may be used.

LACH
Load A, Character

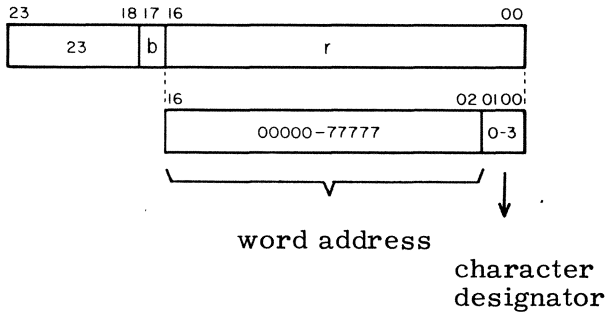


If b = 1, r is modified by index register B^1; R = r + (B^1).
If b = 0, r is not modified (r = R).

Instruction Description: Load bits 00 through 05 of A with the character from storage specified by character address R. The A register is cleared prior to the Load operation.

Comments: Indirect addressing may not be used. Since the sign of B^b is extended during character address indexing, the range of characters referenced by indexing is: r ± $16,383_{10}$ characters. Characters are specified in storage as shown under the LQCH instruction below.

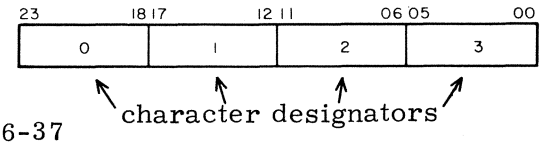
LQCH
Load Q, Character



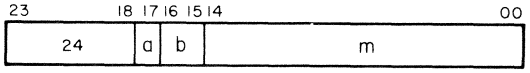
If b = 1, r is modified by index register B^2; R = r + (B^2).
If b = 0, r is not modified (r = R).

Instruction Description: Load bits 00 through 05 of Q with the character from storage specified by character address R. The Q register is cleared prior to the Load operation.

Comments: Indirect addressing may not be used. Since the sign of B^b is extended during character address indexing, the range of characters referenced by indexing is: r ± $16,383_{10}$ characters. Characters are specified in storage as follows:



LCA
Load A, Complement

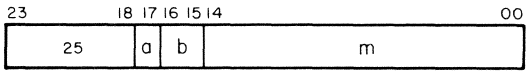


a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Load A with the one's complement of a 24-bit quantity from storage address M.

Comments: Indirect addressing and indexing may be used.

LDAQ
Load AQ

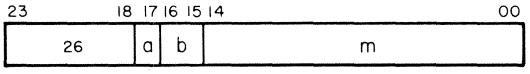


a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Load the A and Q registers with the 24-bit quantities from addresses M and M+1, respectively.

Comments: Addresses 77776 and 77777 should be used only if it is desirable to have M and M+1 as non-consecutive addresses, since one's complement arithmetic is used to form M+1.

LCAQ
Load AQ, Complement

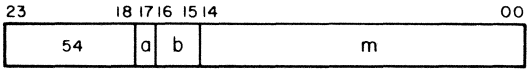


a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Load registers A and Q with the complement of the 24-bit quantities from addresses M and M+1, respectively.

Comments: Addresses 77776 and 77777 should be used only if it is desirable to have M and M+1 as non-consecutive addresses, since one's complement arithmetic is used to form M+1.

LDI
Load Index



a = addressing mode designator
b = index register designator
m = storage address (indexing not permitted)

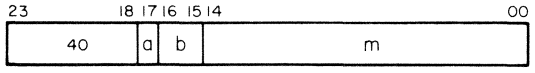
Instruction Description: Load the specified index register, B^b, with the lower 15 bits of the operand stored at address 'm'.

Comments: Indirect addressing may be used, but indexing is not possible anywhere within the indirect address. If indexing in the indirect address is specified, it is ignored. During indirect addressing only 'a' and 'm' are inspected. Symbol 'b' from the initial instruction specifies which index register is to be loaded with the lower 15 bits from the storage address.

Store Instructions

Operation Field		Address Field	Interpretation
STA, I	40	m, b	Store A
STQ, I	41	m, b	Store Q
SACH	42	r, 2	Store A, Character
SQCH	43	r, 1	Store Q, Character
SWA, I	44	m, b	Store Word Address
STAQ, I	45	m, b	Store AQ
SCHA	46	m, b	Store Character Address
STI, I	47	m, b	Store Index

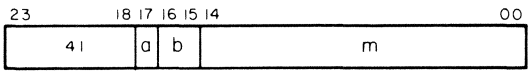
STA
Store A



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Store (A) at the storage address specified by M. The (A) remain unchanged.

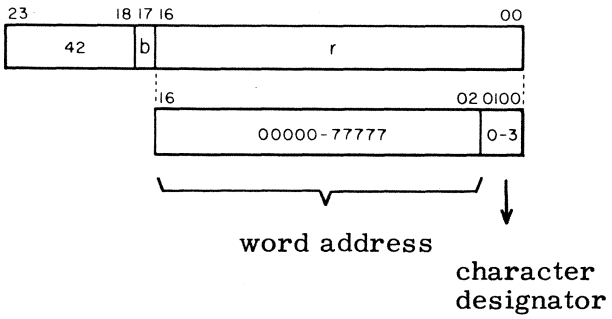
STQ
Store Q



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Store (Q) at the storage address specified by M. The (Q) remain unchanged.

SACH
Store A, Character

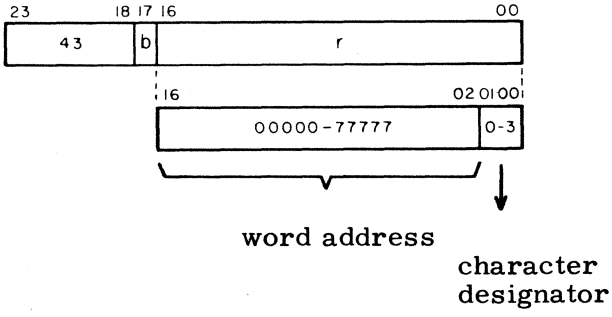


If b = 1, 'r' is modified by index register B^2; R = 'r' + (B^2).
If b = 0, 'r' is not modified (r = R).

Instruction Description: Store the contents of bits 00 through 05 of the A register in the specified character address. All of (A) and the remaining three characters in storage remain unchanged.

Comments: Indirect addressing may not be used. Since the sign of B^b is extended during character address indexing, the range of characters referenced by indexing is: r ± $16,383_{10}$ characters. Characters are specified in storage as shown under the SQCH instruction below.

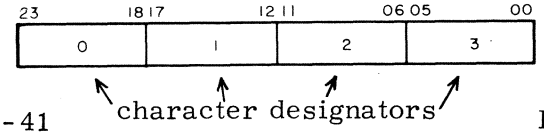
SQCH
Store Q, Character



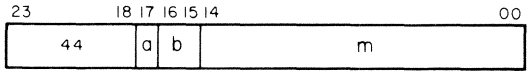
If b = 1, 'r' is modified by index register B^1; R = 'r' + (B^1).
If b = 0, 'r' is not modified (r = R).

Instruction Description: Store the contents of bits 00 through 05 of the Q register in the specified character address. All of (Q) and the remaining three characters in storage remain unchanged.

Comments: Indirect addressing may not be used. Since the sign of B^b is extended during character address indexing, the range of characters referenced by indexing is: r ± $16,383_{10}$ characters. Characters are specified in storage as follows:



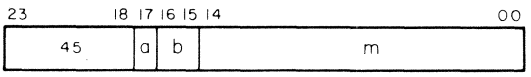
SWA
Store Word Address



a = addressing mode designator
b = index register designator
m = storage address; ($M = m + (B^b)$)

Instruction Description: Store the lower 15 bits of (A) in the designated address M. The upper 9 bits of M and all of (A) remain unchanged.

STAQ
Store AQ

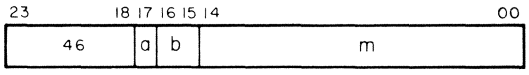


a = addressing mode designator
b = index register designator
m = storage address; $M = m + (B^b)$

Instruction Description: Store (A) and (Q) in the storage locations specified by address M and M+1, respectively. The (A) and (Q) remain unchanged.

Comments: Address 77776 and 77777 should be used only if it is desirable to have M and M+1 as non-consecutive addresses, since one's complement arithmetic is used to form M+1. Indexing and indirect addressing may be used.

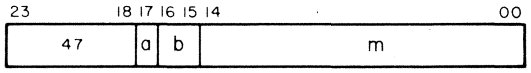
SCHA
Store
Character Address



a = addressing mode designator
b = index register designator
m = storage address; $M = m + (B^b)$

Instruction Description: Store the lower 17 bits of (A) in the address designated by M. The upper 7 bits of M and all of (A) remain unchanged.

STI
Store Index



a = addressing mode designator
b = index register designator
m = storage address (indexing not permitted)

Instruction Description: Store the contents of the specified index register, B^b, in the lower 15 bits of storage address 'm'. The upper 9 bits of 'm' and all of (B^b) remain unchanged.

Comments: Indirect addressing may be used, but indexing is not possible anywhere within the indirect address. If indexing in the indirect address is specified, it is ignored. During indirect addressing only 'a' and 'm' are inspected. The 'b' designator from the initial instruction specifies the index register that will have its contents stored. If 'b' = 0, zeros are stored in the lower 15 bits of 'm'.

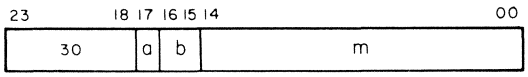
Arithmetic Instructions

Operation Field		Address Field	Interpretation
ADA, I	30	m, b	Add to A
SBA, I	31	m, b	Subtract from A
ADAQ, I	32	m, b	Add to AQ
SBAQ, I	33	m, b	Subtract from AQ
RAD, I	34	m, b	Replace add
MUA, I	50	m, b	Multiply A
DVA, I	51	m, b	Divide A
MUAQ, I	56	m, b	Multiply AQ
DVAQ, I	57	m, b	Divide AQ
FAD, I	60	m, b	FP addition to AQ
FSB, I	61	m, b	FP subtraction from AQ
FMU, I	62	m, b	FP multiplication of AQ
FDV, I	63	m, b	FP division of AQ

NOTE

Indexing and indirect addressing are applicable to all instructions in this group. Reference the Business Data Processing instruction group for the ADM and SBM instructions.

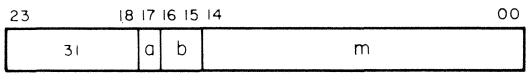
ADA
Add to A



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Add the 24-bit operand located at address M to (A). The sum replaces the original (A).

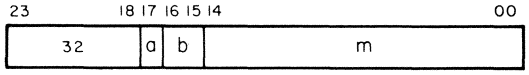
SBA
Subtract from A



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Subtract the 24-bit operand located at address M from (A). The difference replaces the original (A).

ADAQ
Add to AQ

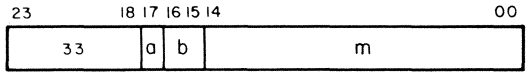


a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Add the 48-bit operand located in addresses M and M+1 to (AQ). The sum is displayed in AQ.

Comments: The upper 24 bits of the 48-bit operand in memory are contained at address M.

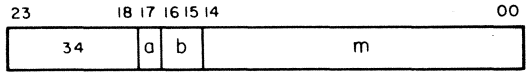
SBAQ
Subtract from AQ



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Subtract the 48-bit operand located in addresses M and M+1 from (AQ). The difference is displayed in AQ.

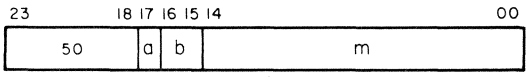
RAD
Replace Add



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Replace the 24-bit operand at address M with the sum of (M) and (A). The original (A) remain unchanged.

MUA
Multiply A



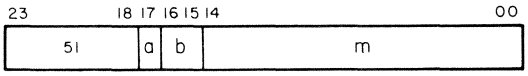
a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Multiply (A) by the operand located at address M.

NOTE

The 48-bit product is displayed in QA. The higher order bits are in Q and the lower order bits are in A.

DVA
Divide A

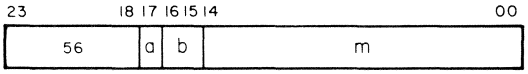


a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Divide the 48-bit operand in AQ by the operand at storage address M. The lower 23 bits of A contain the quotient; the upper bit of A holds the sign. The remainder with sign extended is placed in Q.

Comments: If a divide fault occurs (quotient exceeds 23 bits), the divide operation is terminated and program execution advances to the next address. The result in A and Q are then meaningless.

MUAQ
Multiply AQ

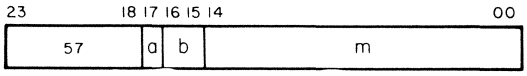


a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Multiply (AQ) by the 48-bit operand in addresses M and M+1. The 96-bit product is displayed in AQE.

Comments: Refer to Figure 6-5 for operand formats. This instruction is unconditionally trapped if the FP MODE switch on the console is OFF.

DVAQ
Divide AQ



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Divide (AQE) by the 48-bit operand in addresses M and M+1. The lower 47 bits of AQ contain the quotient; the upper bit of AQ holds the sign. The remainder with sign extended is placed in E.

Comments: If a divide fault occurs (quotient exceeds 47 bits), program execution advances to the next address. The final contents of AQ and E are then meaningless. Refer to Figure 6-5 for operand formats. This instruction is unconditionally trapped if the FP MODE switch on the console is OFF.

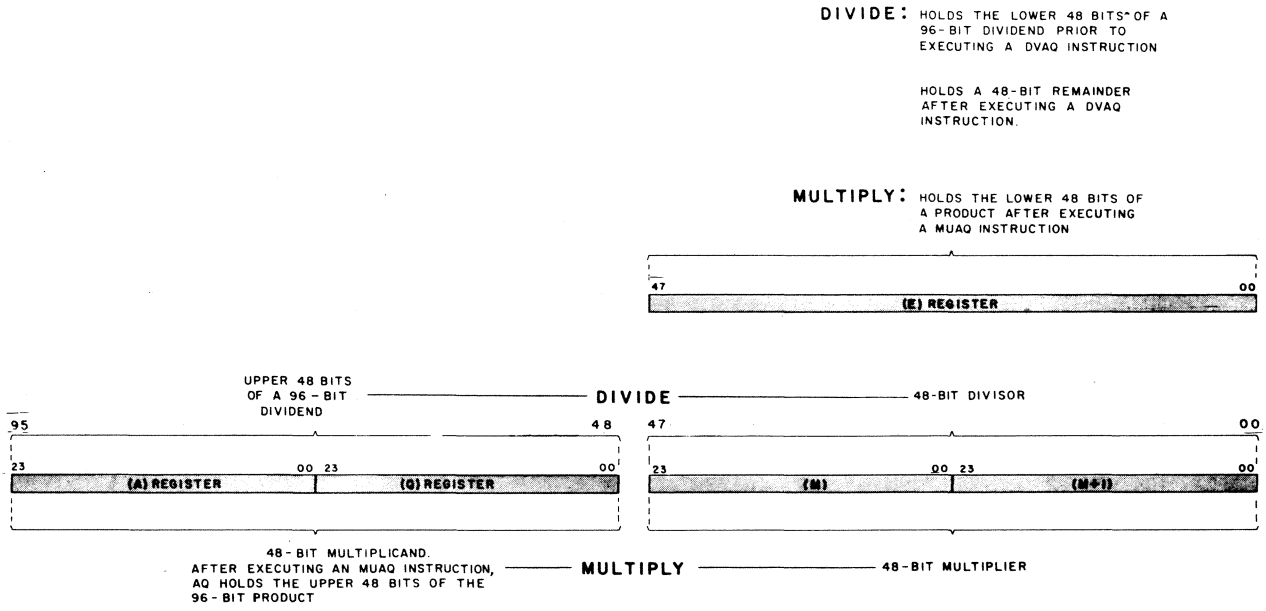
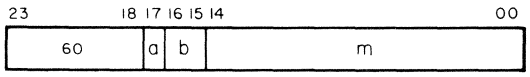


Figure 6-5. Operand Formats and Bit Allocations for MUAQ and DVAQ Instructions

NOTE

The instructions on this page are unconditionally trapped if the FP MODE switch on the console is OFF.

FAD
FP Addition to AQ



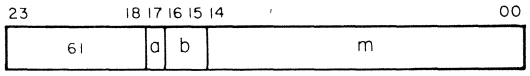
a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Add the 48-bit operand located in addresses M and M+1 to (AQ). The rounded and normalized sum is displayed in AQ.

Comments: The higher order bits of E hold the portion of the operand that was shifted out of AQ during exponent equalization with a maximum of 77_8 shifts being possible. Exponent overflow/underflow fault occurs if the exponent of the sum is not within ± 1777_8.

Refer to Figure 6-6 for operand formats. Supplementary arithmetic information is included in Appendix B.

FSB
FP Subtraction from AQ



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Subtract the 48-bit floating point operand located at storage addresses M and M+1 from the floating point operand in AQ. The rounded and normalized difference is displayed in AQ.

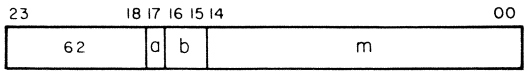
Comments: The higher order bits of E hold the portion of the operand that was shifted out of AQ during exponent equalization with a maximum of 77_8 shifts being possible. Exponent overflow/underflow fault occurs if the exponent of the sum is not within ± 1777_8.

Refer to Figure 6-6 for operand formats. Supplementary arithmetic information is included in Appendix B.

NOTE

The instructions on this page are unconditionally trapped if the FP MODE switch on the console is OFF.

FMU
FP Multiplication of AQ



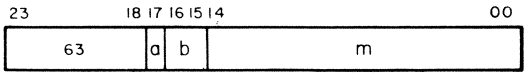
a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Multiply the 48-bit floating point operand in AQ by the floating point operand located at storage addresses M and M+1. The rounded and normalized product is displayed in AQ.

Comments: Bits 12-47 of E hold the lower 36 bits of the 72-bit unnormalized product. Exponent overflow/underflow fault occurs if the exponent of the product is not within ± 1777_8.

Refer to Figure 6-6 for operand formats. Supplementary arithmetic information is included in Appendix B.

FDV
FP Division of AQ



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Divide the floating point operand in AQ by the 48-bit floating point operand located at storage addresses M and M+1. The rounded and normalized quotient is displayed in AQ. The remainder with sign extended appears in the E register.

Comments: The sign of the remainder is the same as that of the dividend. Exponent overflow/underflow occurs if the exponent of the quotient is not within ± 1777_8. The divisor must be properly normalized or a divide fault will result.

Refer to Figure 6-6 for operand formats. Supplementary arithmetic information is included in Appendix B.

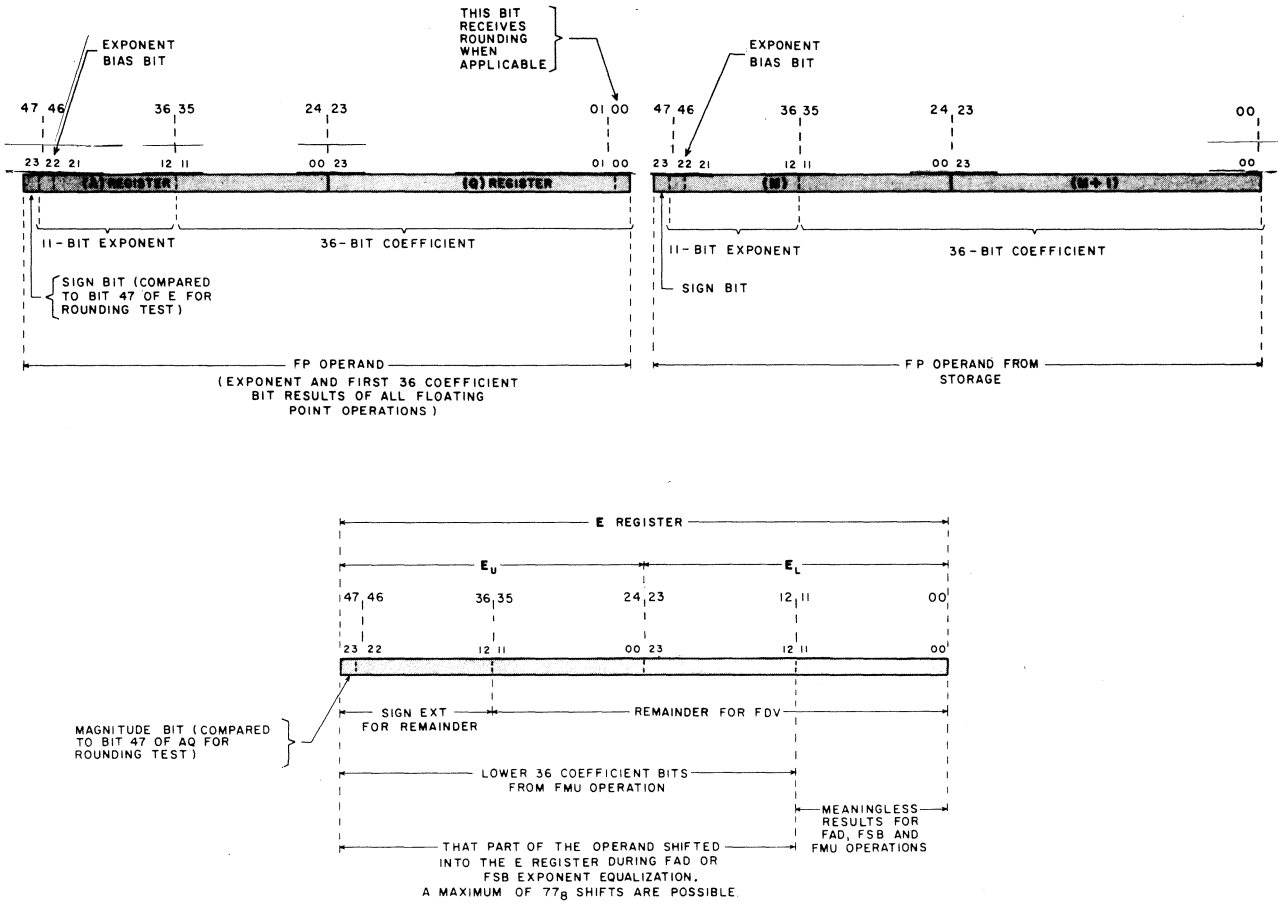


Figure 6-6. Operand Formats and Bit Allocations for Floating Point Arithmetic Instructions

Logical Instructions

Operation Field		Address Field	Interpretation
XOI	16.(0.3)	y, b	Exclusive OR of index and 'y'
XOA	16.6	y	Exclusive OR of A and 'y'
XOA, S	16.4	y	Exclusive OR of A and 'y', sign of 'y' extended
XOQ	16.7	y	Exclusive OR of Q and 'y'
XOQ, S	16.5	y	Exclusive OR of Q and 'y', sign of 'y' extended
ANI	17.(0-3)	y, b	AND of index and 'y'
ANA	17.6	y	AND of A and 'y'
ANA, S	17.4	y	AND of A and 'y', sign of 'y' extended
ANQ	17.7	y	AND of Q and 'y'
ANQ, S	17.5	y	AND of Q and 'y', sign of 'y' extended
LDL, I	27	m, b	Load A with AND of Q and M
SSA, I	35	m, b	Selectively set A
SCA, I	36	m, b	Selectively complement A
LPA, I	37	m, b	Logical product A

The following two examples are logical instructions and illustrate the Exclusive OR and AND functions:

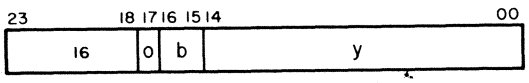
EXAMPLE A:

		(Binary Equivalents)
(A) = 23456701	=	010 011 100 101 110 111 000 001
Execute: 16 4 50321	=	111 111 111 101 000 011 010 001
(XOA, S)		101 100 011 000 110 100 010 000
Final (A)	=	5 4 3 0 6 4 2 0

EXAMPLE B:

(Q) = 23456701	=	010 011 100 101 110 111 000 001
Execute: 17 7 77170	=	000 000 000 111 111 001 111 000
(ANQ)		000 000 000 101 110 001 000 000
Final (Q)	=	0 0 0 5 6 1 0 0

XOI
Exclusive OR of B^b and y

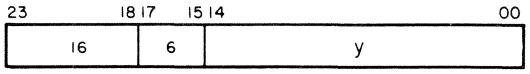


b = index register designator

Instruction Description: Replace the (B^b) with the selective complement (the Exclusive OR function) of 'y' and (B^b).

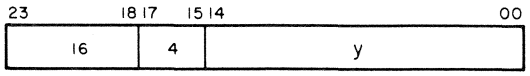
Comments: If 'b' = 0, this is a No-Operation instruction. If the optional RIP is present, RIP MODE selected, and execution takes place in Monitor state of Executive mode, 'b' = 0 denotes one of eight RIP instructions rather than a No-Op. Refer to the Real-Time Interrupt Instructions.

XOA
Exclusive OR of A and y



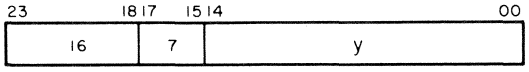
Instruction Description: Replace the (A) with the selective complement (the Exclusive OR function) of 'y' and (A).

XOA, S
Exclusive OR of A and y, Sign Extended



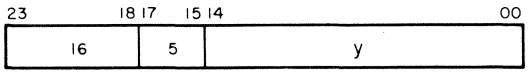
Instruction Description: Same as XOA except the sign of 'y' is extended.

XOQ
Exclusive OR of Q and y



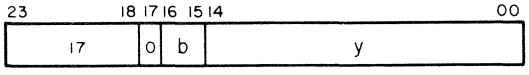
Instruction Description: Replace the (Q) with the selective complement (the Exclusive OR function) of 'y' and (Q).

XOQ, S
Exclusive OR of Q and y, Sign Extended



Instruction Description: Same as XOQ except the sign of 'y' is extended.

ANI
AND of B^b and y

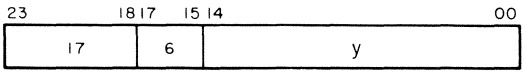


b = index register designator

Instruction Description: Replace the (B^b) with the logical product (the AND function) of 'y' and(B^b).

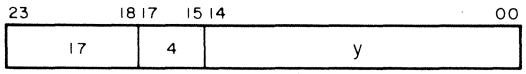
Comments: If 'b' = 0, this is a No-Operation instruction. If the optional RIP is present, RIP MODE selected, and execution takes place in Monitor state of Executive mode, 'b' = 0 denotes a Restore (RST) or a Restore and Interrupt (RSTI) instruction rather than a No-Op. Refer to the Real-Time Interrupt Instructions.

ANA
AND of A and y



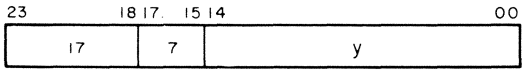
Instruction Description: Replace the (A) with the logical product (the AND function) of 'y' and (A).

ANA, S
AND of A and y,
Sign Extended



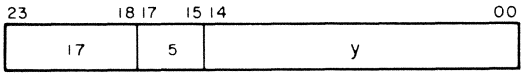
Instruction Description: Same as ANA except the sign of 'y' is extended.

ANQ
AND of Q and y



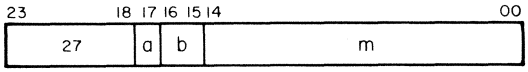
Instruction Description: Replace the (Q) with the logical product (the AND function) of 'y' and (Q).

ANQ, S
AND of Q and y,
Sign Extended



Instruction Description: Same as ANQ except the sign of 'y' is extended.

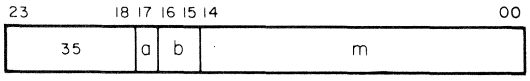
LDL
Load A, Logical



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Load A with the logical product (the AND function) of (Q) and the 24-bit quantity from storage address M.

SSA
Selectively Set A



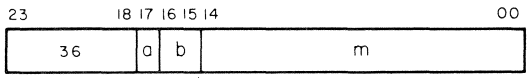
a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Selectively set the bits of the A register to "1's" for all corresponding "1's" in the quantity at address M. Initial "1's" in A remain unchanged.

EXAMPLE:
(A) = 23456710 = 010 011 100 101 110 111 001 000
(M) = 76345242 = 111 110 011 100 101 010 100 010

Final (A) = 111 111 111 101 111 111 101 010
7 7 7 5 7 7 5 2

SCA
Selectively Complement A



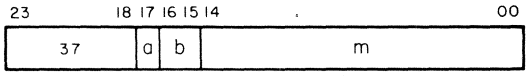
a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Selectively complement the bits in the A register that correspond to the "1" bits in the quantity at address M.

EXAMPLE:
(A) = 23456710 = 010 011 100 101 110 111 001 000
(M) = 20341573 = 010 000 011 100 001 101 111 011

Final (A) = 000 011 111 001 111 010 110 011
0 3 7 1 7 2 6 3

LPA
Logical Product A



a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: Replace (A) with the logical product of (A) and (M).

EXAMPLE:
(A) = 23456710 = 010 011 100 101 110 111 001 000
(M) = 45210376 = 100 101 010 001 000 011 111 110

Final (A) = 000 001 000 001 000 011 001 000
0 1 0 1 0 3 1 0

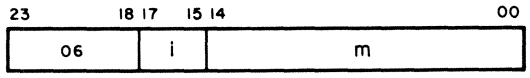
Search, Move, and Compare Instructions

Operation Field		Address Field	Interpretation
MEQ	06	m, i	Masked equality search
MTH	07	m, i	Masked threshold search
SRCE	71	c, r, s	Search for character equality
SRCN	71	c, r, s	Search for character inequality
MOVE	72	$, r, s	Move characters from 'r' to 's'
CPR	52	m, b	Compare (within limits test)

NOTE

Related instructions are included in the Business Data Processing group.

MEQ Masked Equality Search



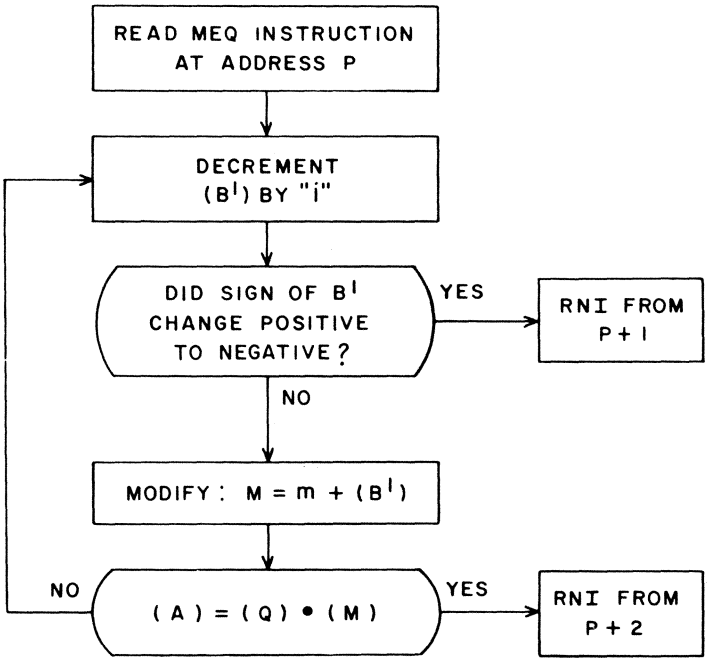
i = interval designator, 0 to 7
m = storage address

Instruction Description: (A) is compared with the logical product of (Q) and (M). This instruction uses index register B^1 exclusively. 'm' is modified just prior to step 3 in the test below. Instruction sequence follows:

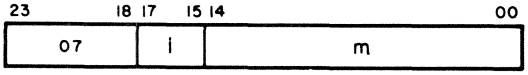
1. Decrement (B^1) by 'i'. (Refer to table below.)
2. If (B^1) changed sign from positive to negative, RNI from P + 1; if not,
3. Test to see if (A) = (Q) • (M). M = 'm' + (B^1).
   If (A) = (Q) • (M), RNI from P + 2; if not,
4. Repeat the sequence.

Comments: 'i' is represented by 3 bits, permitting a decrement interval selection from 1 to 8. Positive zero and negative zero are recognized as equal quantities. A test is made for active interrupt conditions during the ROP sequence for this instruction. If an interrupt condition is active, an instruction is again read from memory after which the condition is recognized.

DESIGNATOR i	DECREMENT INTERVAL
1	1
2	2
3	3
4	4
5	5
6	6
7	7
0	8


READ MEQ INSTRUCTION AT ADDRESS P
DECREMENT (B') BY "i"
DID SIGN OF B' CHANGE POSITIVE TO NEGATIVE?
YES
RNI FROM P+1
NO
MODIFY: M = m + (B')
NO
(A) = (Q) • (M)
YES
RNI FROM P+2

MTH
Masked
Threshold Search



i = interval designator, 0 to 7
m = storage address

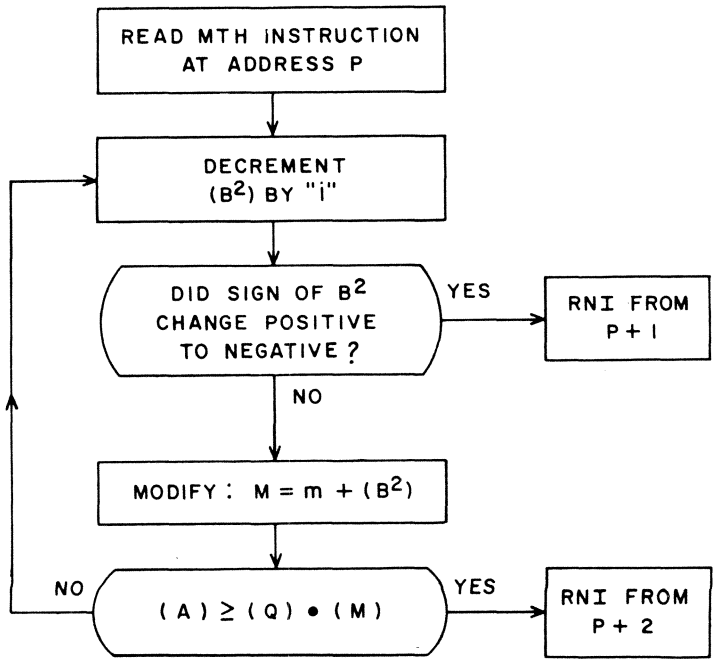
Instruction Description: (A) is compared with the logical product of (Q) and (M). This instruction uses index register B^2 exclusively. 'm' is modified just prior to step 3 in the test below.

Instruction Sequence:

1. Decrement (B^2) by 'i'. (Refer to table below.)
2. If (B^2) changed sign from positive to negative, RNI from P + 1; if not,
3. Test to see if (A) $\geq$ (Q) $\bullet$ (M). M = m + (B^2).
   If (A) $\geq$ (Q) $\bullet$ (M), RNI from P + 2; if not.
4. Repeat the sequence.

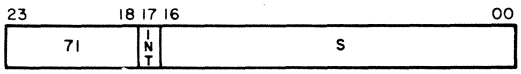
Comments: 'i' is represented by 3 bits, permitting a decrement interval selection from 1 to 8. Positive zero and negative zero are recognized as equal quantities. A test is made for active interrupt conditions during the ROP sequence for this instruction. If an interrupt condition is active, an instruction is again read from memory after which the condition is recognized.

DESIGNATOR i	DECREMENT INTERVAL
1	1
2	2
3	3
4	4
5	5
6	6
7	7
0	8

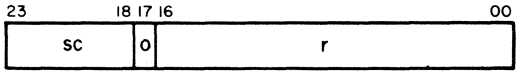


SRCE
Search for Character Equality

P



P+1



INT = '1' for interrupt upon completion
s = last character address of the search block, plus one
sc = 6-bit BCD scan character
r = first (current) character address of the search block

Instruction Description: This instruction initiates a search through a block of character addresses in storage looking for equality with the scan character, 'sc'. If Search/Move control is not busy, the buffered search operation commences while Main Control performs a RNI at P + 3. Main Control continues executing the main program while the search operation occurs simultaneously. If Search/Move control is initially busy, Main Control performs a RNI at P + 2 and the search operation does not occur.

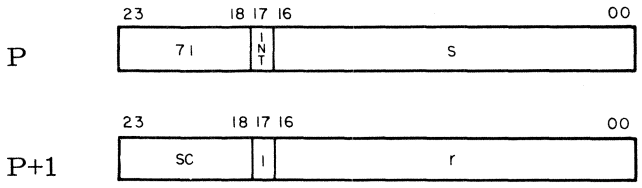
As a search progresses, 'r' is incremented until the search terminates when either a comparison occurs between the scan character 'sc' and a character in the storage field, or until 'r'='s'. If a comparison does occur, the address of the satisfying character may be determined by inspecting 'r'. To do this, transfer the contents of register 20 to A with instruction TMA (53 0 20020).

Register 20 of the register file is reserved for the second instruction word which contains the current character address of the search block. Register 30 is reserved for the first instruction word which contains the last character address of the search block, plus one.

Figure 6-7 is a flow chart of steps that occur during a SRCE operation.

Comments: Before executing this instruction in Executive mode, the desired program state number (0 through 7) must be loaded into the lower three bits of the A register. This number is then automatically transferred to the upper 3 bits of Register File location 20 for referencing during the buffered operation.

SRCN
Search for Character Inequality



INT = "1" for interrupt upon completion
s = last character address of the search block, plus one
sc = 6-bit BCD scan character
r = first (current) character address of the search block

Instruction Description: This instruction initiates a search through a block of character addresses in storage looking for inequality with scan character 'sc'. If Search/Move control is not busy, the buffered search operation occurs simultaneously. If Search/Move control is initially busy, Main Control performs an RNI at P + 2 and the search operation does not occur.

As a search progresses, 'r' is incremented until the search terminates when either an unequal character comparison occurs between the scan character 'sc' and a character in storage, or until 'r' = 's'. If an unequal character comparison does occur, the address of the satisfying character may be determined by inspecting 'r'. To do this, transfer the contents of Register 20 to A with instruction TMA (53 0 20020).

Register 20 of the register file is reserved for the second instruction word which contains the current character address of the search block. Register 30 is reserved for the first instruction word which contains the last character address of the search block, plus one.

Figure 6-7 is a flow chart of steps that occur during a SRCN operation.

Comments: Before executing this instruction in Executive mode, the desired program state number (0 through 7) must be loaded into the lower three bits of the A register. This number is then automatically transferred to the upper 3 bits of Register File location 20 for referencing during the buffered operation.

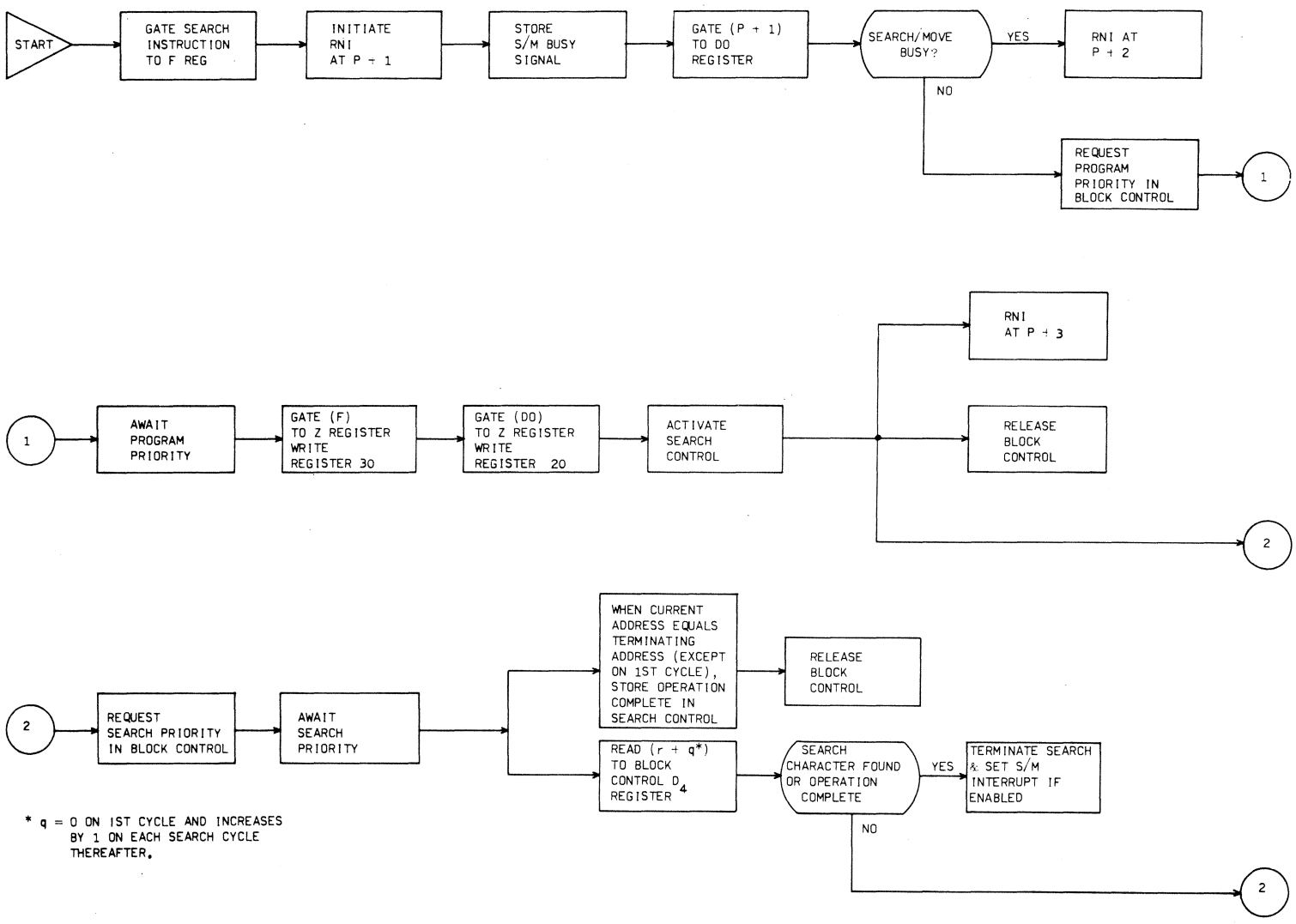
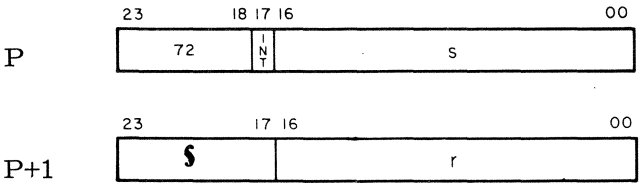


Figure 6-7. SRCE and SRCN (71) Operation

MOVE
Move § Characters from r to s



INT = '1' for interrupt upon completion
s = first address of character block destination
§ = field length of data block, 0-177$_8$* characters
r = first address of character block source

Instruction Description: This instruction moves a block of characters from one area of storage to another. If Search/Move control is not busy, the buffered move operation commences while Main Control performs a RNI at P + 3. Main Control continues executing the main program while the move operation occurs simultaneously. If Search/Move control is initially busy, Main Control performs a RNI at P + 2 and the move operation does not occur.

As a move operation progresses, 'r' and 's' are incremented and § (number of characters) is decremented until § = 0. 128 characters or 32 words may be moved. When bits 00 and 01 of 'r' and 's' are zero, and the field length is a multiple of four characters, data is moved word by word. This reduces the move time by 75 percent over a character by character

Register 21 of the Register File is reserved for the second instruction word which contains the first address of the character block source. Register 31 is reserved for the first instruction word which contains the first address of the character block destination.

Figure 6-8 is a flow chart of steps that occur during a MOVE operation.

Comments: Before executing this instruction in Executive mode, the desired program state number (0 through 7) must be loaded into the lower three bits of the A register. This number is then automatically transferred to the upper 3 bits of Register File location 21 for referencing during the buffered operation.

*= 1-177$_8$ represents a field length of 1 to 127 characters; 0 represents a field length of 128 characters.

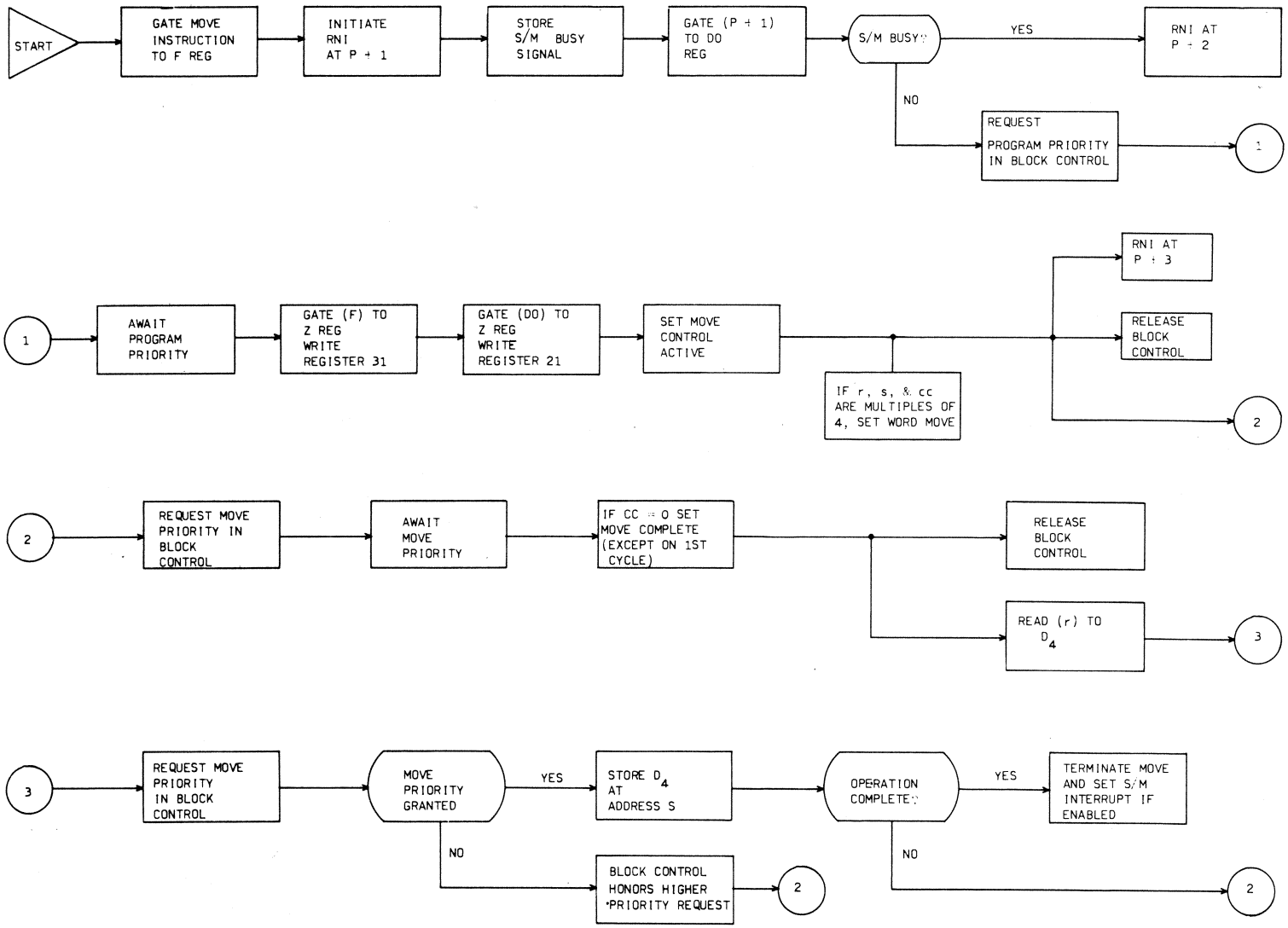
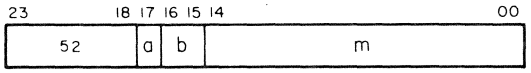


Figure 6-8. MOVE (72) Operation

CPR Compare (Within Limits Test)



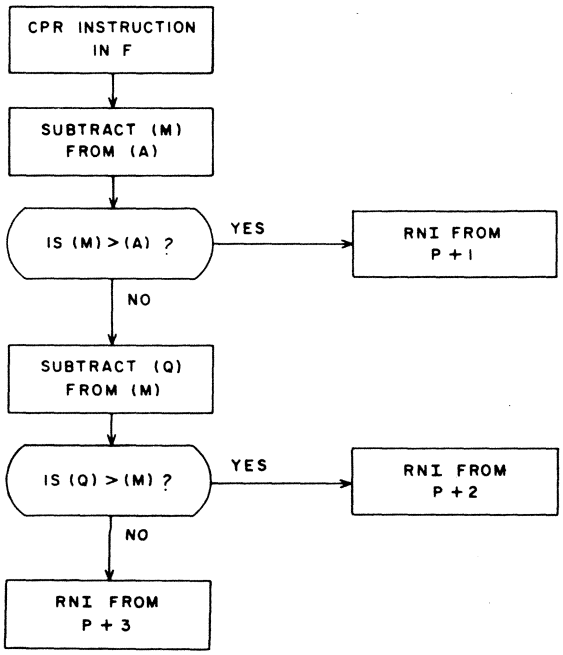
a = addressing mode designator
b = index register designator
m = storage address; M = m + (B^b)

Instruction Description: The quantity stored at address M is tested to see if it is within the upper limits specified by A and the lower limits specified by Q. The testing proceeds as follows:

1. Subtract (M) from (A). If (M) > (A), RNI from address P + 1; if not,
2. Subtract (Q) from (M). If (Q) > (M), RNI from P + 2; if not,
3. RNI from address P + 3.

Comments: The original state of the A and Q registers remains unchanged. (A) must be $\geq$ (Q) initially or the test cannot be satisfied. 77777777 is not sensed as negative zero. The following table is a synopsis of the CPR test:

Test Sequence	Jump Address if Test is Satisfied
(M) > (A)	P + 1
(Q) > (M)	P + 2
(A) $\geq$ (M) $\geq$ (Q)	P + 3



CPR FLOW CHART

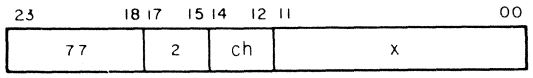
Sensing Instructions

Operation Field		Address Field	Interpretation
EXS	77.2	x, ch; x ≠ 0	*Sense external status
COPY	77.2	x, ch; x = 0	Copy external status
INS	77.3	x, ch; x ≠ 0	*Sense internal status
CINS	77.3	x, ch; x = 0	Copy internal status
INTS	77.4	x, ch	*Sense interrupt

NOTE

An Executive interrupt is generated if an attempt is made to execute any of these instructions during Program State of Executive mode. Refer to the ACI instruction for special considerations regarding the 'ch' designator in these instructions.

EXS
Sense
External Status



ch = I/O channel designator, 0-7
x = external status sensing mask code (see Comments below)

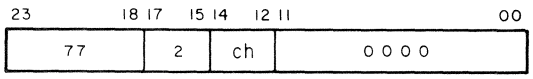
Instruction Description: When a peripheral equipment controller is connected to an I/O channel by the CON (77.0) instruction, the EXS instruction can sense conditions within that controller. Twelve status lines run between each controller and its I/O channel. External status lines for missing channels are sensed as being active. Each line may monitor one condition within the controller, and each controller has a unique set of line definitions.

To sense a specific condition, a "1" is placed in the bit position of the status sensing mask that corresponds to the line number. When this instruction is recognized, RNI at address P + 1 if an external status line is active when its corresponding mask bit is "1". If no selected line is active, RNI at address P + 2.

Comments: Refer to the 3000 Series Computer Systems Peripheral Equipment Codes manual for a complete list of status response codes.

*The Disable Advance P Switch will not prevent the P Register from advancing to P+1 or P+2.

COPY
Copy
External Status



ch = I/O channel designator, 0-7

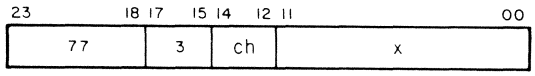
Instruction Description: This instruction performs the following functions:

1. The external status code from I/O channel 'ch' is loaded into the lower 12 bits of A. (See EXS instruction.)
2. The contents of the Interrupt Mask register are loaded into the upper 12 bits of A. (See Table 6-2.)
3. RNI from address P + 1.

TABLE 6-2. INTERRUPT MASK REGISTER BIT ASSIGNMENTS

Masked Bit Positions	Mask Codes (x)	Interrupt Conditions Represented
00	0001	I/O Channel 0 (includes interrupt gener-
01	0002	1 ated within the channel
02	0004	2 and external equipment
03	0010	3 interrupts)
04	0020	4
05	0040	5
06	0100	6
07	0200	7
08	0400	Real-time clock
09	1000	Exponent overflow/underflow & BCD faults
10	2000	Arithmetic overflow and divide faults
11	4000	Search/Move completion

INS
Sense Internal Status



ch = I/O channel designator, 0-7
x = internal status sensing mask code

Instruction Description: Table 6-3 lists the bit definitions of the internal status sensing mask. Bits 00-04 and 06-07 represent conditions within I/O channel 'ch'. Internal status lines for missing channels are sensed as being active. Bits 05 and 08-11, which represent internal faults, may be sensed without regard to channel designation.

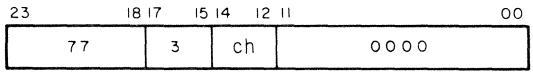
To sense a specific condition, load a "1" into the bit position of the mask that corresponds to the condition. When this instruction is executed, RNI from address P + 1 if an internal status line is active and the corresponding mask bit is a "1". If none of the selected lines are active, RNI from address P + 2.

TABLE 6-3. INTERNAL STATUS SENSING MASK

Masked Bit Positions	Mask Codes (x)	Interrupt Conditions Represented
00	0001	Parity error on channel 'ch'
01	0002	Channel 'ch' busy reading
02	0004	Channel 'ch' busy writing
03	0010	External reject active on channel 'ch'
04	0020	No-response reject active on channel 'ch'
05	0040	*Illegal write
06	0100	Channel 'ch' preset by CON or SEL, but no reading or writing in progress
07	0200	Internal I/O channel interrupt on channel 'ch' upon: 1) completion of read or write operation, or 2) end of record
08	0400	*Exponent overflow/underflow fault (floating point)
09	1000	*Arithmetic overflow fault (adder)
10	2000	*Divide fault
11	4000	*BCD fault

*Internal faults are cleared when sensed.

CINS
Copy Internal Status

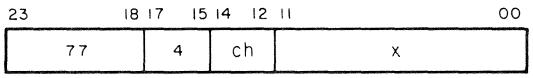


ch = I/O channel designator, 0-7

Instruction Description: This instruction performs the following functions:

1. The internal status code is loaded into the lower 12 bits of A. (See INS instruction.)
2. The contents of the Interrupt Mask register are loaded into the upper 12 bits of A. (See Table 6-2.)
3. RNI from address P + 1.

INTS
Sense Interrupt



ch = I/O channel designator, 0-7
x = interrupt sensing mask code

Instruction Description: Sense for the interrupt conditions listed in Table 6-4. RNI from P + 1 if an interrupt line is active and the corresponding sensing mask bit is a "1". If none of the selected lines are active, RNI from P + 2. Bits 08-11 represent conditions that may be sensed without regard to channel designation. External interrupt lines for missing channels are sensed as being active.

TABLE 6-4. INTERRUPT SENSING MASK

Masked Bit Positions	Mask Codes (x)	Interrupt Conditions Represented
00	0001	External equipment interrupt line 0 active
01	0002	1
02	0004	2
03	0010	3
04	0020	4
05	0040	5
06	0100	6
07	0200	7
08	0400	*Real-time clock
09	1000	*Exponent overflow/underflow & BCD faults
10	2000	*Arithmetic overflow & divide faults
11	4000	*Search/Move completion
*Internal faults are cleared when sensed.		

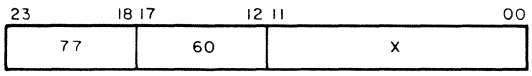
Pause Instructions

Operation Field	Address Field	Interpretation
PAUS 77.60	x	*Pause on condition
PRP 77.61	x	*Priority pause

NOTE

An Executive interrupt is generated if an attempt is made to execute either of these instructions during Program state of Executive mode.

PAUS
Pause



x = pause sensing mask code

Instruction Description: This instruction allows the program to halt for a maximum of 40 ms if a condition defined by the pause sensing mask (Table 6-5) exists (see the NOTE below for exceptions). If a "1" appears on a line that corresponds to a mask bit that is set, the count in P will not advance. If the advancement of P is delayed for more than 40 ms, the next instruction is read from address P + 1. If none of the lines being sensed are active, or if they become inactive during the pause, the program immediately skips to address P + 2. If an interrupt occurs and is enabled during a PAUS, the pause condition is terminated, the interrupt sequence is initiated, and the address of the PAUS instruction is stored as the interrupt address.

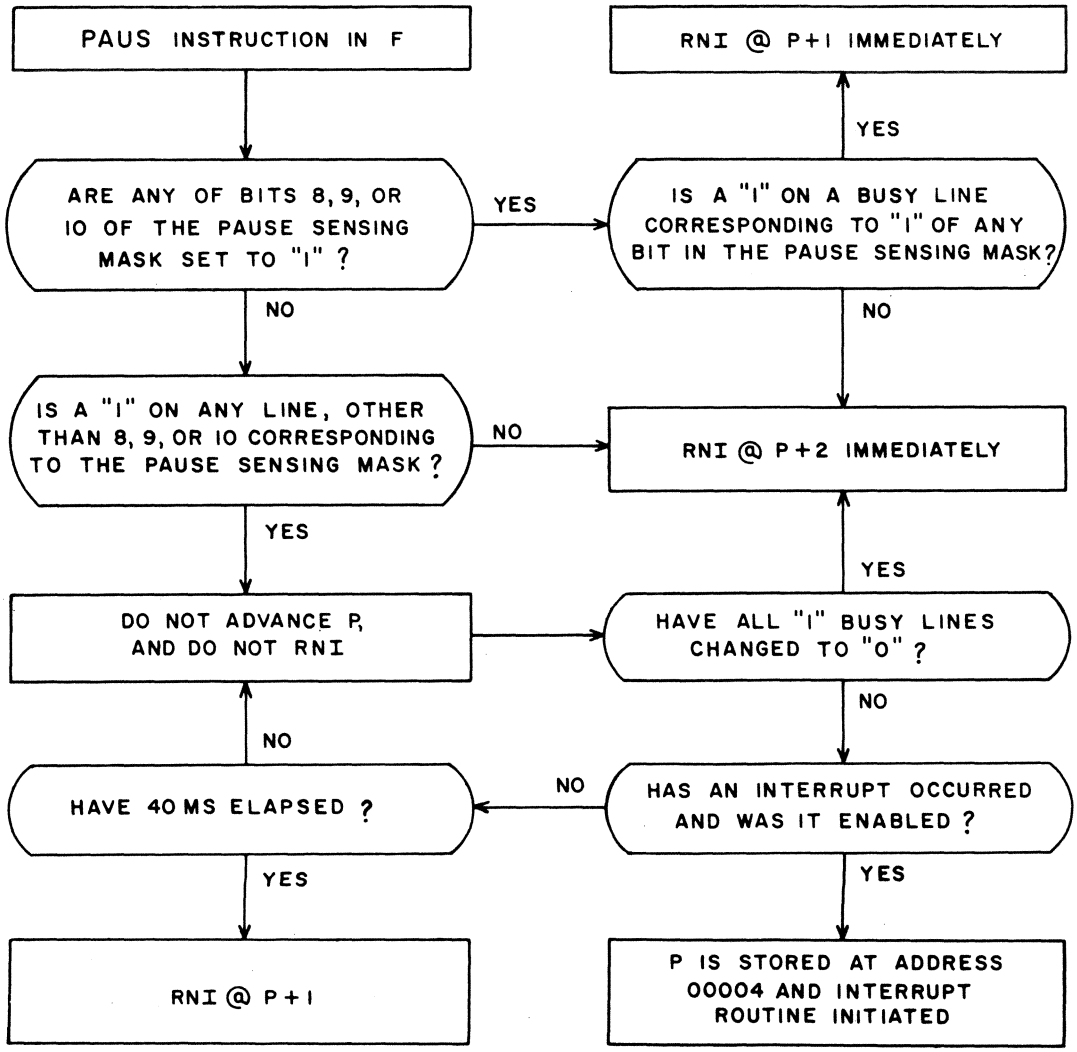
NOTE

If bit 08, 09, or 10 (or any combination of these bits) is set, the PAUS actually becomes a sense instruction. If any condition defined by the mask exists, a RNI is immediately made from P+1. If no defined conditions are active, a skip exit is made to P+2. TYPE FINISH and/or TYPE REPEAT are cleared if bit 09 and/or bit 10 of the mask are set and the condition(s) does not exist.

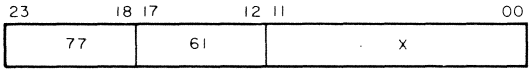
*The Disable Advance P Switch will not prevent the P Register from advancing to P+1 or P+2.

TABLE 6-5. PAUSE SENSING MASK

Mask Bits	Mask Codes	Condition	Notes
00	0001	I/O channel 0 busy	Channel read or write operation in progress, the External MC logic within the channel is set, or a Reply or Reject from a previous operation is still present at the channel
01	0002	1	
02	0004	2	
03	0010	3	
04	0020	4	
05	0040	5	
06	0100	6	
07	0200	7	
08	0400	Typewriter busy	Typewriter I/O in progress
09	1000	Typewriter NOT finish	Finish logic not set
10	2000	Typewriter NOT repeat	Repeat logic not set
11	4000	Search/Move control busy	Search or Move operation in progress



PRP
Priority Pause



x = pause sensing mask code

Instruction Description: This instruction performs the same operation as the preceding PAUS (77.6) instruction, however, the real-time clock is prevented from incrementing during the pause.

Comments: Preventing the real-time clock from incrementing enables Block Control to continue an I/O operation without being referenced by the clock. This provides a more efficient I/O transfer operation.

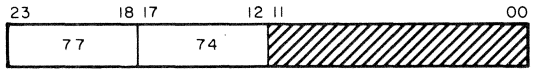
Interrupt Instructions

Operation Field		Address Field	Interpretation
EINT	77.74		Enable interrupt control
DINT	77.73		Disable interrupt control
SSIM	77.52	x	Selectively set interrupt mask
SCIM	77.53	x	Selectively clear interrupt mask
INCL	77.50	x	Clear interrupt
CILO	77.511	cm	Channel interrupt lockout
IAPR	77.57		Interrupt associated processor
SFPF	77.71		Set floating point fault
SBCD	77.72		Set BCD fault

NOTE

An Executive interrupt is generated if an attempt is made to execute any of these instructions (except SFPF and SBCD) during Program state of Executive mode.

EINT
Enable
Interrupt Control

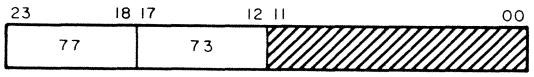


Bits 00 - 11 should be loaded with zeros.

Instruction Description: This instruction enables the CPU interrupt control system, and records the fact by setting bit 04 of the Condition register.

Comments: One additional instruction at P + 1 is executed before the processor can be interrupted, provided that the additional instruction requires no more than one RADR cycle. If the EINT instruction is executed at P, the earliest possible interrupt will occur at P + 2. If the instruction following the EINT contains more than one RADR cycle, the earliest interrupt can occur during its second RADR cycle.

DINT
Disable
Interrupt Control

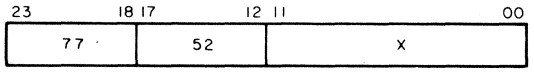


Bits 00 - 11 should be loaded with zeros.

Instruction Description: This instruction immediately disables the CPU interrupt control system and clears bit 04 of the Condition register. Interrupts are not recognized during the RNI for this instruction.

Comments: The interrupt system remains disabled until an EINT instruction is executed. Selected interrupts may still be sensed.

SSIM
Selectively Set
Interrupt Mask Register



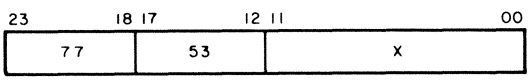
x = interrupt mask register codes

Instruction Description: This instruction selectively sets the Interrupt Mask register according to the interrupt mask code 'x'. For each bit set to "1" in x, the corresponding bit position in the Interrupt Mask register is set to "1" (see Table 6-6). Bit positions representing non-existent I/O channels cannot be set.

Comments: A program test for the existence of I/O channels for a system is as follows:

1. Set the interrupt mask bits to all "1's" by executing a SSIM (77 5 27777) instruction.
2. Execute either a COPY or CINS instruction and examine the upper 12 bits of A.
3. As bits representing non-existent I/O channels cannot be set, a "0" in bits 00-07 of the Interrupt Mask register indicates a non-existent I/O channel.

SCIM
Selectively Clear
Interrupt Mask Register



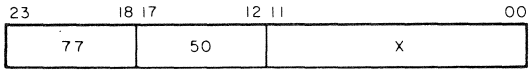
x = interrupt mask register codes

Instruction Description: This instruction selectively clears the Interrupt Mask register according to the interrupt mask code 'x'. For each bit set to "1" in 'x', the corresponding bit position in the Interrupt Mask register is set to "0" (see Table 6-6).

TABLE 6-6. INTERRUPT MASK REGISTER BIT ASSIGNMENTS

Mask Bit Positions	Mask Codes (x)	Interrupt Conditions Represented
00	0001	I/O Channel 0 (includes interrupts gen-
01	0002	1 erated within the channel
02	0004	2 and external equipment
03	0010	3 interrupts)
04	0020	4
05	0040	5
06	0100	6
07	0200	7
08	0400	Real-time clock
09	1000	Exponent overflow/underflow & BCD faults
10	2000	Arithmetic overflow and divide faults
11	4000	Search/Move completion

INCL
Clear Interrupt

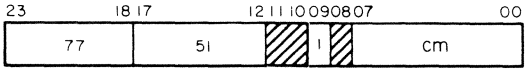


x = interrupt mask register codes

Instruction Description: This instruction clears the interrupt faults defined by the mask codes in Table 6-6. I/O channel interrupts are cleared by this instruction and although the Interrupt Clear is sent to peripheral equipment, not all equipments drop their interrupt lines. Refer to the Peripheral Equipment Reference Manual for information on specific equipment.

Comments: All external interrupts are locked out for 1 usec after execution of this instruction.

CILO
Channel Interrupt Lockout

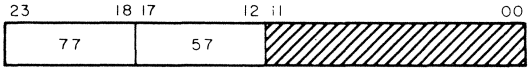


cm = channel mask
Bits 08, 10, and 11 should be loaded with zeros.

Instruction Description: Disables all external interrupts on channel(s) 'cm' while the channel(s) are busy. Termination of the I/O operation clears the disabling function. When necessary, peripheral equipment can override this interrupt lockout.

Comments: Bit 00 corresponds to channel 0, bit 01 corresponds to channel 1 etc. More than one channel may be set to "1" for multiple channel interrupt lockout.

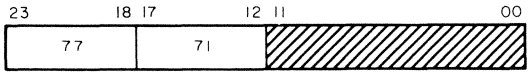
IAPR
Interrupt
Associated Processor



Bits 00 - 11 should be loaded with zeros.

Instruction Description: The processor (computer) executing this instruction sends an interrupt to an associated processor via a special cable. The interrupt remains active in the receiving computer until it is recognized.

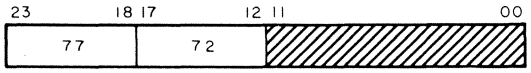
SFPF
Set Floating
Point Fault



Bits 00 - 11 should be loaded with zeros.

Instruction Description: This instruction sets the floating point fault flag in the central processor.

SBCD
Set BCD Fault



Bits 00 - 11 should be loaded with zeros.

Instruction Description: This instruction sets the BCD fault flag in the central processor.

Comments: Normally, the BCD fault flag is set when a BCD fault occurs during the execution of one of the BDP (64-70) instructions. This instruction allows interpretive software to set the fault when processing trapped BCD instructions (BDP MODE not selected).

Real-Time Interrupt Instructions

Operation Field		Address Field	Interpretation
CRI	15.0		Clear RIP
CRIM	16.00	g, x	Selectively clear real-time interrupt mask
SRIM	16.01	g, x	Selectively set real-time interrupt mask
SIBA	16.02	d	Set interrupt base address
FRI	16.03	g, 1	Force real-time interrupt
DRIP	16.04		Disable RIP
ERIP	16.05		Enable RIP
SAD	16.06		Select automatic disable
DAD	16.07		Deselect automatic disable
RST	17.00	1	Restore
RSTI	17.04	1	Restore and interrupt

NOTE

All RIP programming must be done in Monitor state of Executive mode.

The Real-Time Interrupt instructions are used in conjunction with the optional RIP. These instructions are No-Ops if: the RIP is not present, the RIP MODE switch is disabled, the CPU is in the Non-Executive mode, or the instruction is executed in Program state of Executive mode.

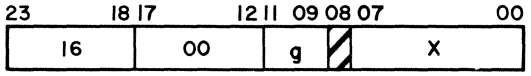
CRI
Clear Real-Time Interrupt Processor

Bits 00-14 should be loaded with zeros.

Instruction Description: This instruction clears all recorded and/or recognized interrupts in the RIP, the RIP enable, and the automatic disable feature.

Comments: The Interrupt Mask register and the Interrupt Base Address register are not altered by this instruction.

CRIM
Selectively Clear Real-Time Interrupt Mask



Bit 08 should be a zero.

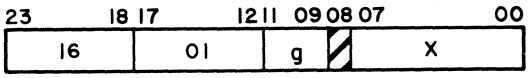
g = interrupt group designator (0-7).

x = interrupt mask code (8 mask bits per group).

Instruction Description: This instruction selectively clears the RIP interrupt mask according to the group selected by 'g' and the mask code 'x for that group. For example, the instruction 16001002 clears the second mask bit in group 1 (the tenth of the 64 mask bits). See table below.

Mask Bits	Group (g)
7 6 5 4 3 2 1 0	↓
8 ——— 1	0
16 ——— 9	1
24 ——— 17	2
32 ——— 25	3
40 ——— 33	4
48 ——— 41	5
56 ——— 49	6
64 ——— 57	7

SRIM
Selectively Set Real-Time Interrupt Mask



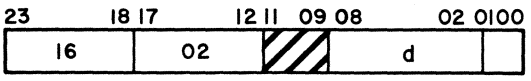
Bit 08 should be a zero.

g = interrupt group designator (0 - 7)

x = interrupt mask code (8 mask bits per group)

Instruction Description: This instruction selectively sets the RIP interrupt mask according to the group selected by 'g' and the mask code 'x' for that group. For example, the instruction 16015377 sets all the mask bits in group 5 (the 41st through the 48th of the 64 mask bits). See the table following the CRIM instruction.

SIBA
Set Interrupt Base Address



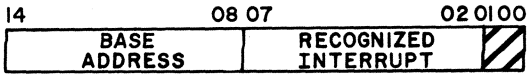
Bits 00, 01, and 09 - 11 should be loaded with zeros.

d = interrupt base address designator

Instruction Description: This instruction sets up the base address portion of the interrupt address according to the interrupt base address designator 'd'.

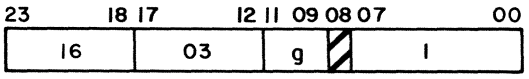
Comments: The base address is the upper 7 bits of the RIP interrupt address as shown below. Refer to Real-Time Interrupts in Section 4.

Interrupt Address



Bits 00 and 01 should be zero.

FRI
Force
Real-Time Interrupt

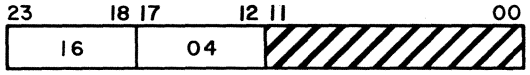


Bit 08 should be a zero.
g = interrupt group designator (0 - 7)
1 = interrupt line code (8 interrupt lines per group)

Instruction Description: This instruction forces the activation of interrupts in the RIP according to the group selected by 'g' and the interrupt line code '1' for that group.

Comments: The forced interrupts are used for maintenance or diagnostic purposes. All considerations for recognizing normal RIP interrupts apply to forced interrupts: the corresponding mask bit must be set, no higher priority interrupts may be active, and the RIP must be enabled.

DRIP
Disable Real-Time
Interrupt Processor

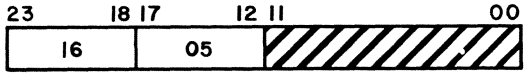


Bits 00 - 11 should be loaded with zeros.

Instruction Description: This instruction disables the RIP by preventing interrupt recognition until an ERIP instruction is executed.

Comments: The RIP is also disabled by an internal clear, a Master clear, or an abnormal interrupt.

ERIP
Enable Real-Time
Interrupt Processor

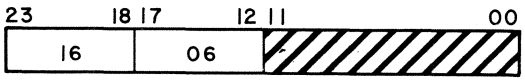


Bits 00 - 11 should be loaded with zeros.

Instruction Description: This instruction enables the RIP, allowing interrupt recognition to occur.

Comments: The first ERIP instruction following an abnormal interrupt will not enable the RIP if the RIP was disabled at the time the abnormal interrupt occurred; the second ERIP instruction following an abnormal interrupt will unconditionally enable the RIP.

SAD
Select
Automatic Disable

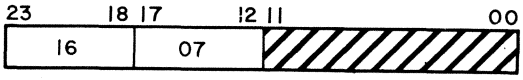


Bits 00 - 11 should be loaded with zeros.

Instruction Description: This instruction allows the RIP to be disabled automatically when any real-time interrupt occurs.

Comments: Additional real-time interrupts cannot be recognized until the ERIP instruction is executed. The automatic disable condition is in effect until a DAD instruction, an internal clear, or a Master clear is executed.

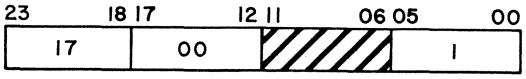
DAD
Deselect
Automatic Disable



Bits 00 - 11 should be loaded with zeros.

Instruction Description: This instruction clears the automatic disable condition in the RIP which is set by an SAD instruction.

RST
Restore

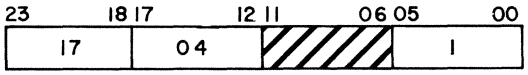


Bits 06 - 11 should be loaded with zeros.
1 = interrupt line designator

Instruction Description: This instruction signals the RIP that interrupt line '1' and all interrupt lines lower in priority may be recognized. If interrupt line '1' is specified as one of lines 00- 17_8, this instruction also signals the RIP to restore the operational registers stored at the time the interrupt on line '1' occurred.

Comments: All real-time interrupts with the exception of the RIP Parity Error are disabled between the execution of the RST instruction and the next Jump instruction. Refer to Real-Time Interrupts in Section 4 for additional information.

RSTI
Restore and Interrupt



Bits 06 - 11 should be loaded with zeros.
1 = interrupt line designator

Instruction Description: This instruction performs the same restore function as the RST instruction. In addition, it causes a normal Interrupt signal in the CPU providing the Real-Time Interrupt Flag is set.

Comments: The Real-Time Interrupt Flag sets upon recognition of a Real-Time Interrupt signal from the RIP and is cleared upon recognition of any normal interrupt including the Restore interrupt.

When the Restore Interrupt is recognized, a 0317 code is stored in the lower 12 bits of address 00005. Refer to Normal Interrupt Recognition in Section 4 for additional information.

Program and Relocation Control Instructions

Operation Field		Address Field	Interpretation
RIS	55.0		Relocate with (ISR)
ROS	55.4		Relocate with (OSR)
JAA	77.56		Transfer Last Jump Addr to A
TMAV	77.61		Test Memory Availability
SBJP	77.62		Set boundary jump flag
SDL	77.624		Set destructive load flag
CRA	77.63		Transfer (CR) to A_{00-05}
ACR	77.634		Transfer A_{00-05} to CR
APF	77.64	w, 2	Transfer A to Page File
PFA	77.65	w, 2	Transfer Page File to A
AOS	77.66		Transfer (A_{00-02}) to OSR
AIS	77.664		Transfer (A_{00-02}) to ISR
OSA	77.67		Transfer (OSR) to A_{00-02}
ISA	77.674		Transfer (ISR) to A_{00-02}

NOTE

All instructions in this group are No-Ops during Non-Executive mode. All instructions except RIS and ROS cause an Executive interrupt if their execution is attempted during Program state.

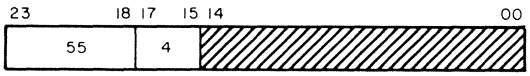
RIS
Relocate with Instruction State

Bits 00 - 14 should be loaded with zeros.

Instruction Description: Clear bit 02 of the Condition register, enabling (ISR) to be used as the upper 3 bits of address for all storage references during Program state.

Comments: A Master Clear produces the same effect as this instruction. Refer to Section 5 for additional programming information.

ROS
Relocate with Operand State

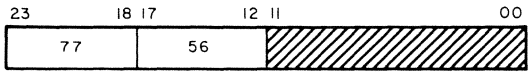


Bits 00 - 14 should be loaded with zeros.

Instruction Description: Set bit 02 of the Condition register, enabling (OSR) to be used as the upper 3 bits of address for all operand references in Executive mode. Blue background light of OSR register display will also be on.

Comments: The Condition register is cleared upon execution of the CRA instruction used in interrupt routines. Refer to Section 5 for additional programming information.

JAA
Jump Address → A

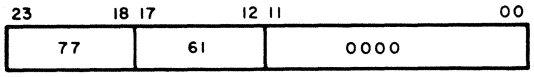


Bits 00 - 11 should be loaded with zeros.

Instruction Description: Transfer the address, P, of the last jump type instruction occurring in Program state, to A00-14.

Comments: The Last Jump Address (LJA) can also be displayed on the console when the LJA switch is pressed and the computer is stopped (refer to Section 7).

TMAV
Test Memory Availability

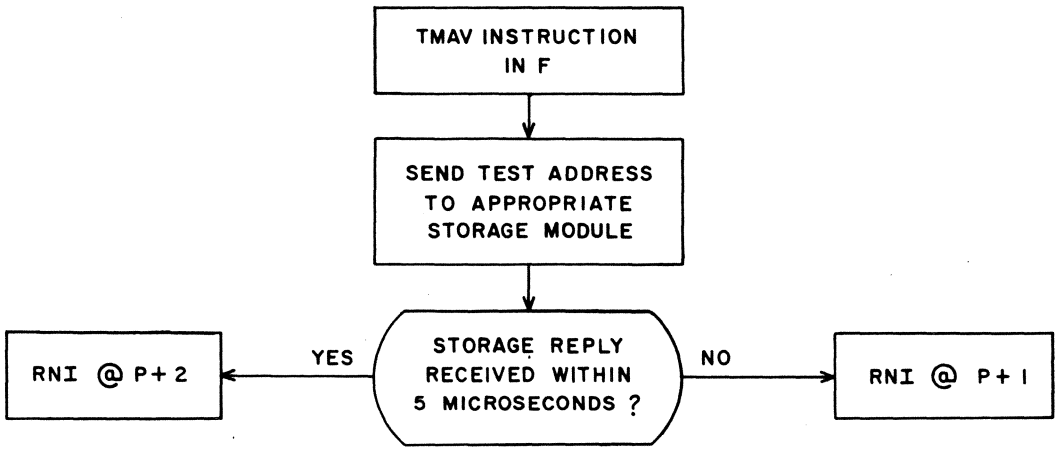


Instruction Description: This instruction is used to test core storage for the presence of a particular address. If a storage reply is received within 5 microseconds after executing the instruction, the address does exist, and the next instruction is read from P+2. If a reply does not occur within 5 microseconds, the address does not exist in the system and the next instruction is read from P+1. The contents of the test address are of no significance.

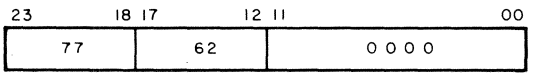
Comments: Prior to executing this instruction, the lower 15 bits of the testing address must be formed in B^2. The upper 3 bits of the address will be zeros unless the OSR has been previously selected by the ROS (55.4) instruction.

This instruction does not cause a Storage Parity Error - No Response interrupt.

See the flow chart on the following page.



SBJP Set Boundary Jump

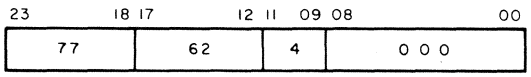


Instruction Description: Set bit 00 of the Condition register to flag a Boundary Jump condition.

Comments: When the next jump instruction is executed, the boundary jump condition flag is cleared and the processor reverts to Program state. The (ISR) are appended to the jump address (m) for the RNI that follows. The (ISR) are used for both the STO and the RNI if the jump instruction is an RTJ (00.7).

If the computer is interrupted, the condition is cleared as the CRA instruction used in interrupt processing is executed.

SDL
Set
Destructive Load



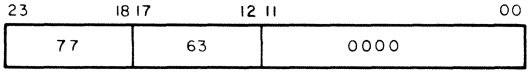
Instruction Description: Set bit 01 of the Condition register to flag a Destructive Load condition.

Comments: After this instruction is executed, the next Load A (LDA) instruction senses the flag and causes the following operations to occur:

1. Load (M) from LDA instruction into A and restore 77777777 into (M).
2. Clear the destructive load condition flag when executing the LDA instruction.

If the computer is interrupted, this condition is cleared as the CRA instruction used in interrupt processing is executed.

CRA
(CR) → $A_{00\text{-}05}$



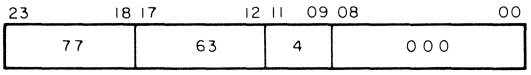
Instruction Description: Transfer the (CR) to the lower 6 bits of A. The Condition register is cleared upon completion of the transfer.

Comments: This instruction should be used in interrupt routines during Executive Mode operations. All interrupts are disabled between interrupt recognition and the execution of the instruction. Program conditions represented by the Condition register are listed below:

Bit 00 - Boundary Jump
Bit 01 - Destructive Load A
Bit 02 - Operand Relocation Using OSR
Bit 03 - Program State
Bit 04 - Interrupt System Enabled
Bit 05 - Program State

The (CR) are automatically transferred to the RIP Register File upon interrupting on real-time interrupt lines 00 - 17_8. The Condition register is cleared upon completion of the transfer. The CRA instruction is not required in this case, but is required for real-time interrupts on lines 20_8 - 77_8. Refer to Real-Time Interrupts in Section 4 for additional information.

ACR
$(A_{00-05}) \rightarrow CR$

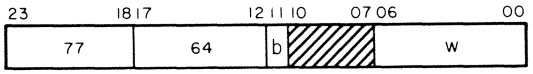


Instruction Description: Transfer the (A_{00-05}) to the Condition register.

Comments: This instruction should be used at the end of interrupt routines during Executive mode to restore the Condition register to its original state. All interrupts are disabled between the execution of the ACR instruction and the jump instruction used to exit from interrupt routines. Refer to the CRA instruction for the conditions represented in the Condition register.

The RST (17.00) or RSTI (17.04) instruction must be used rather than the ACR instruction for real-time interrupt routines on lines 00 - 17_8. Refer to Real-Time Interrupts in Section 4 for additional information.

APF
$(A_{00-11}) \rightarrow PIF$

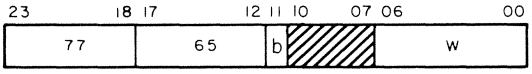


b = index designator, B^2 only
Bits 07 - 10 should be loaded with zeros.

Instruction Description: Transfer (A_{00-11}) to the 12-bit index at Page Index File address 'w'.

Comments: If bit 11 is a "1", (B^2) are used for indexing. This instruction is a No-Op if the optional address relocation feature is not present.

PFA
$(PIF) \rightarrow A_{00-11}$

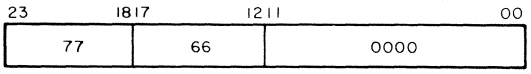


b = index designator, B^2 only
Bits 07 - 10 should be loaded with zeros.

Instruction Description: Transfer the 12-bit index at Page Index File address 'w' to A_{00-11}.

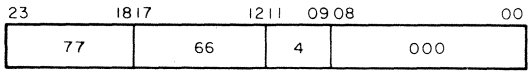
Comments: If bit 11 is a "1", (B^2) are used for indexing. This instruction is a No-Op if the optional address relocation feature is not present.

AOS
$(A_{00-02}) \rightarrow OSR$



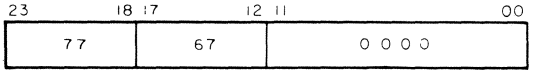
Instruction Description: Transfer (A_{00-02}) to the Operand State register.

AIS
$(A_{00-02}) \rightarrow ISR$



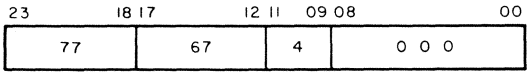
Instruction Description: Transfer (A_{00-02}) to the Instruction State register.

OSA
$(OSR) \rightarrow A_{00-02}$



Instruction Description: Transfer (OSR) to A_{00-02}.

ISA
$(ISR) \rightarrow A_{00-02}$



Instruction Description: Transfer (ISR) to A_{00-02}.

Input/Output Instructions

Operation Field		Address Field	Interpretation
ACI	77.54		Transfer (A_{00-02}) to CIR
CIA	77.55		Transfer (CIR) to A_{00-02}
IOCL	77.51	x	Clear I/O channel(s) and equipment
CLCA	77.512	cm	Clear I/O channel(s)
CON	77.0	x, ch	Connect to external equipment
SEL	77.1	x, ch	Select function
CTI	77.75		Set console typewriter input
CTO	77.76		Set console typewriter output
INPC, INT B, H, G	73	ch, r, s	Character-addressed input to storage
INPW, INT, B, N, G	74	ch, m, n	Word-addressed input to storage
INAC, INT, H	73	ch	Input character to A
INAW, INT	74	ch	Input word to A
OUTC, INT, B, H	75	ch, r, s	Character-addressed output from storage
OUTW, INT, B, N	76	ch, m, n	Word-addressed output from storage
OTAC, INT, H	75	ch	Output character from A
OTAW, INT, B, N	76	ch	Output word from A

NOTE

An Executive interrupt is generated if an attempt is made to execute any of these instructions during Program state of Executive mode.

Main Control gains access to Block Control to initiate the section for I/0 and Search/Move operations (71 - 76 instructions). This initiation consists of storing modified forms of the instruction words in the Register File, providing addressing and other operating information to Block Control (see Table 6-7). The S/M instructions are described in the Search, Move, and Compare Instruction group.

Registers $00\text{-}17_8$ of the file are reserved for I/O operations. Registers 00-07 are used for storing the modified instruction words containing the current character addresses. Registers $10\text{-}17_8$ hold the modified sub-instruction words containing the last character addresses (plus or minus one, depending upon instruction parameters). Before executing an I/O with storage instruction during Executive mode, the desired 'state' number must be loaded into the lower 3 bits of the A register. This number is then automatically stored in the upper 3 bits of Register File location 0X during initiation of the buffer, and becomes the upper 3 bits of the address sent to the relocation section during servicing of the buffer.

Prior to executing word addressed I/O instructions, the addresses 'm' and 'n' are shifted left two places to form character addresses. These character addresses are stored in the file for use during the buffer execution.

I/O operations with storage and output from A are buffered. After a buffer operation is initiated, buffer requests to Block Control are honored on a priority basis, with Channel 0 having highest priority (see Block Control Priority in Section 1). Each I/O instruction should be referenced for a description of its particular parameters and a flowchart of the overall operation. Descriptions of the I/O channel, interface signals, and the parity checking used in the I/O system are given in Section 3. Attention should be given to the peripherals when programming, since extraneous parity errors may result when doing input operations with a controller that sends more data than requested. The ACI instruction should be referenced for special I/O channel considerations.

Table 6-7 and its accompanying example illustrate the relationship between the Register File addresses, their contents, and the individual instructions.

TABLE 6-7. MODIFIED I/O INSTRUCTION WORDS

	Instruction	Relative location of instruction words (see individual instructions)	Register File location	Contents of Register File location
Operations With Storage	73 (INPC)	P	1X	3 - - - - - - -
		P + 1	0X	* - - - - - - -
	74 (INPW)	P	1X	0 - - - - - - -
		P + 1	0X	* - - - - - - -
	75 (OUTC)	P	1X	1 - - - - - - -
		P + 1	0X	* - - - - - - -
	76 (OUTW)	P	1X	2 - - - - - - -
		P + 1	0X	* - - - - - - -
Operations With A	73 (INAC)	P	1X	7 - - - - - - -
		P + 1	0X	* - - - - - - -
	74 (INAW)	P	1X	4 - - - - - - -
		P + 1	0X	* - - - - - - -
	75 (OTAC)	P	1X	5 - - - - - - -
		P + 1	0X	* - - - - - - -
	76 (OTAW)	P	1X	6 - - - - - - -
		P + 1	0X	* - - - - - - -

**X = I/O Channel ('ch' designator OR'ed with (CIR)-Exec)
('ch' designator only if Non-Executive)

* = 'state' number (contents of ISR or OSR) - Executive Mode

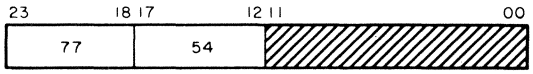
EXAMPLE: Execute the following INPW instruction during Executive mode.

P = 74 003200	(A) = 00000001
P + 1 = 20 003100	(CIR) = 4
P + 2 = 01 003300	

ANALYSIS: I/O Channel 6 is used. Since character addresses are formed, Register File location 16 holds 04015000, and location 06 holds 10014400.

This instruction specifies 12- to 24-bit assembly, no interrupt upon completion, forward storage, and an unconditional jump to address 03300 as a reject instruction.

ACI
$(A_{00-02}) \rightarrow CIR$



Bits 00 - 11 should be loaded with zeros.

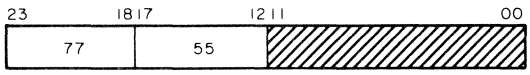
Instruction Description: Transfer (A_{00-02}) to the Channel Index register.

Comments: This instruction is used to set the CIR to a value from 0 to 7. The (CIR) are logically ORed with the 'ch' designator in the following instructions while in Monitor state.

73 - 76	I/O instructions	77.2	Copy External Status
77.0	Connect	77.3	Sense Internal Status
77.1	Select Function	77.3	Copy Internal Status
77.2	Sense External Status	77.4	Sense Interrupt

This instruction is a No-Op if operating in Non-Executive mode.

CIA
$(CIR) \rightarrow A_{00-02}$

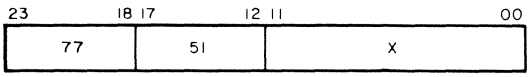


Bits 00 - 11 should be loaded with zeros.

Instruction Description: Transfer (CIR) to A_{00-02}.

Comments: This instruction is a No-Op if operating in Non-Executive mode.

IOCL
Clear I/O, Typewriter, and Search/Move



x = block control clearing mask

Instruction Description: This instruction may be used to clear the I/O channels. It also clears all associated peripheral equipment, the typewriter or the Search/Move control according to bits set in the Block Control clearing mask (see Table 6-8).

Comments: All external interrupts are locked out for 1 usec following execution of this instruction.

TABLE 6-8. BLOCK CONTROL CLEARING MASK

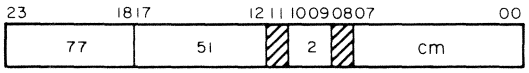
Mask Bits	Mask Codes (x)	Controls Cleared
00	0001	I/O channel 0
01	0002	1
02	0004	2
03	0010	3
04	0020	4
05	0040	5
06	0100	6
07	0200	7
08	0400	Typewriter
09	1000	(see note)
10	2000	(see note)
11	4000	Search/Move

NOTE

If bits 09 and 10 are both set or both clear, the channel(s) specified by bits 00 - 07 of the mask are cleared, i.e., Read or Write, Status, and Channel Interrupt are cleared. Bit 08 clears the typewriter as well as the Type Load or Type Dump logic in Block Control. A 5.5 usec Clear signal is also sent to the peripheral equipment connected to the selected channel(s).

If bit 09 is clear and bit 10 is set, the instruction will clear the channel(s) only and the 5.5 usec Clear signal is not transmitted.

CLCA
Clear Channel Activity

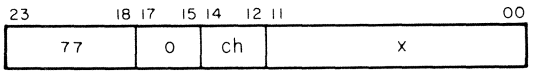


cm = channel mask
Bits 08 and 11 should be loaded with zeros.

Instruction Description: Clear only the selected I/O channel(s).

Comments: The peripheral equipment associated with the selected channel(s) are not cleared by executing this instruction. Bit 00 corresponds to channel 0, bit 01 corresponds to channel 1, etc. More than one channel may be set to "1" for multiple channel clearing.

CON
Connect



ch = I/O channel designator, 0-7
x = 12-bit connect code. Bits 09-11 select one of eight controllers which may be attached to channel ch. Bits 00-08 select the peripheral units connected to the controller.

Instruction Description: This instruction sends a 12-bit connect code along with a connect enable to an external equipment controller on I/O channel ch. If a Reply is received from the controller within 100 usec, the next instruction is read from address P + 2. If a Reject is received or there is no response within 100 usec, or if the I/O channel is busy, a reject instruction is read from address P + 1.

Comments: Refer to Figure 6-9 for a flow chart of the Connect instruction.

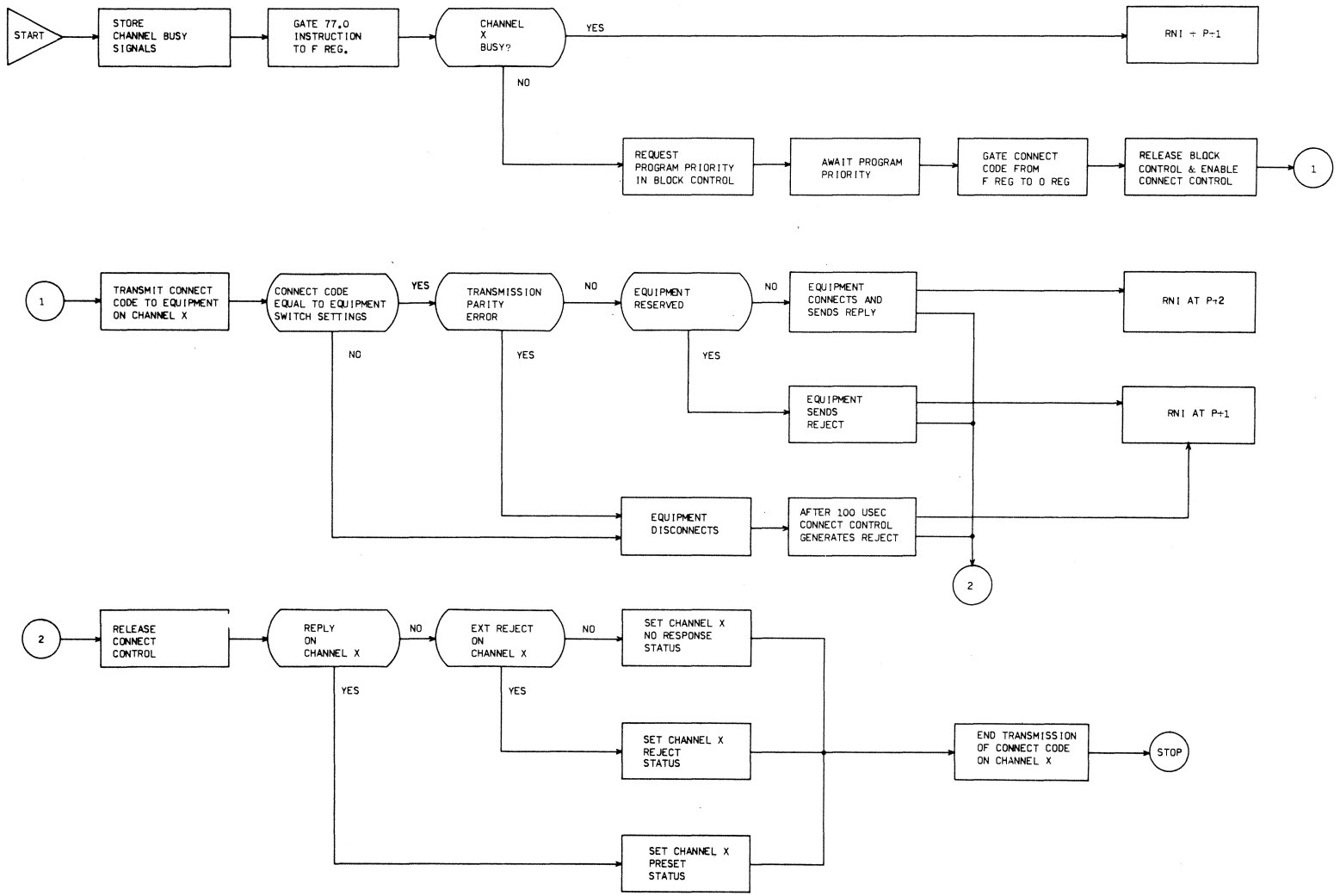
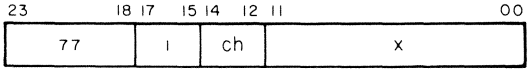


Figure 6-9. Connect (77.0) Operation

SEL
Select Function



ch = I/O channel designator, 0-7
x = 12-bit function code. Each piece of external equipment has a unique set of function codes to specify operations within that device. Refer to the 3000 Series Computer Systems Peripheral Equipment Codes Pub. No. 60113400 for a complete list of function codes.

Instruction Description: This instruction sends a 12-bit function code along with a function enable to the unit connected to I/O channel 'ch'. If a Reply is received from the unit within 100 usec, the next instruction is read from P + 2. If a Reject is received, or there is no response within 100 usec, or if the I/O channel is busy, a reject instruction is read from address P + 1.

The following conditions or combination of conditions result in a Reject:

1. No Unit or Equipment Connected: The referenced device is not connected to the system and cannot recognize a Select Function instruction.
2. Undefined Code: When the function code 'x' is not defined for the specific device, a Reject may be generated by the device. However, in some cases an undefined code will cause the device to generate a Reply although no operation is performed. (Refer to the reference manual pertaining to the specific peripheral device.)
3. Equipment or Unit Busy or Not Ready: The device cannot perform the operation because a previously specified operation is still in progress. For example, a Write End of File code is rejected by a tape unit if the tape unit is rewinding.
4. Channel Busy: The selected data channel is currently performing a Read or Write operation or a Reply or Reject from a previous operation is still present at the channel.

A flow chart for the Select Function instruction is located on the following page.

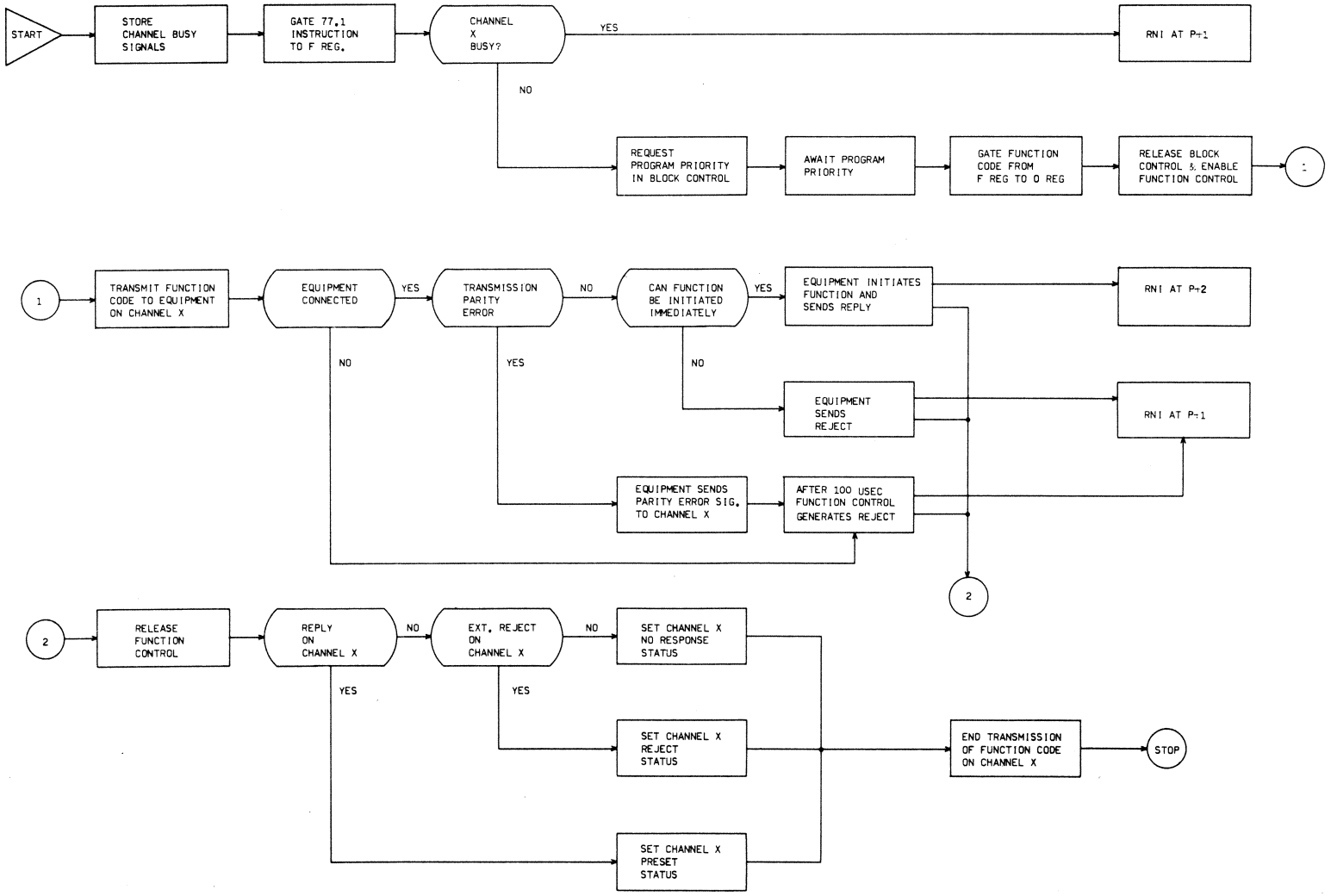
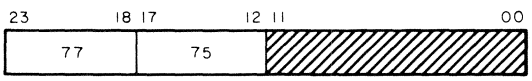


Figure 6-10. Select Function (77.1) Operation

NOTE

The CTI and CTO instructions are mutually exclusive. Typewriter busy should be checked before these instructions are used and before locations 23 and 33 of the Register File are altered.

CTI
Set Console Typewriter Input



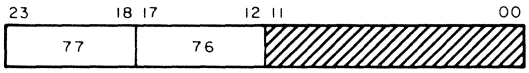
Bits 00 - 11 should be loaded with zeros.

Instruction Description: This instruction performs the same function as pressing the TYPE LOAD switch. A block of data may be entered into storage via the typewriter as soon as the TYPE LOAD indicator lights.

Comments: If a block of data smaller than the one defined by registers 23 and 33 is to be typed, the FINISH switch should be pressed when the typing is completed. If more data is entered than the defined block can hold, the excess data is lost. The REPEAT switch can be pressed to terminate the input operation if a typing error occurs.

FINISH and/or REPEAT can be sensed (and cleared) via status bits 9 and 10 (see PAUS instruction). The formats for registers 23 and 33 are shown on the following page.

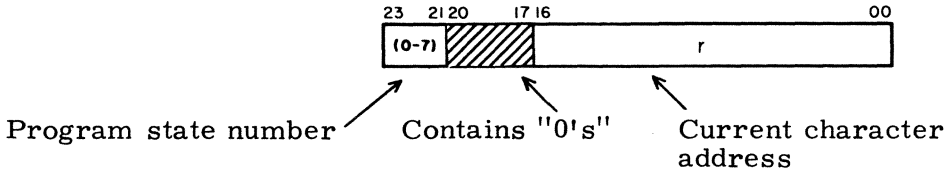
CTO
Set Console Typewriter Output



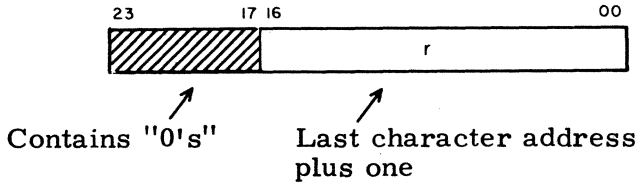
Bits 00 - 11 should be loaded with zeros.

Instruction Description: This instruction, like the TYPE DUMP switch, causes the typewriter to print out the block of data defined by the character addresses in registers 23 and 33. Formats of these register file locations are shown below:

The contents of register 23 should have the following format:

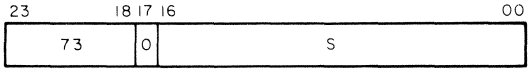


The contents of register 33 should have the following format:

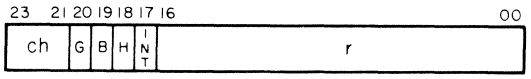


INPC Character-Addressed Input to Storage

P



P+1



B = "1" for backward storage
ch = I/O channel designator, 0-7
H = "0" for 6- to 24-bit assembly
H = "1" for 12- to 24-bit assembly
INT = "1" for interrupt upon completion
r = first character address of I/O data block; becomes current address as I/O operation progresses
s = last character address of input data block, plus one (minus one, for backward storage)
G = "1" for word count control

Instruction Description: This instruction transfers 6-bit characters or 12-bit bytes from an external equipment to storage. If channel 'ch' is not busy the buffered input operation commences while Main Control simultaneously performs a RNI at P + 3. If channel 'ch' is initially busy, Main Control performs a RNI at P + 2 and the input operation does not occur.

Comments: During 6- to 24-bit assembly, 6-bit characters are loaded into successive character locations in storage, beginning with character address 'r'. The next character is loaded into character address 'r + 1' if doing forward storage, or into address 'r - 1' if backward storage is specified.

During 12- to 24-bit assembly, successive 12-bit bytes are loaded into successive halves of storage words. The lowest bit of both character addresses is forced to "0" for the operation, insuring that assembled bytes are stored in either the upper or lower half of a word. To insure transfer of a correct number of characters, an even character count must be specified.

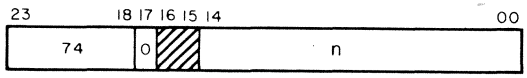
NOTE

When bit 20 of the sub-instruction word at P + 1 is a "1", the word count control feature allows the input operation to continue beyond an End of-Record signal(s).

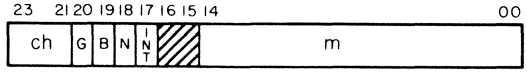
Refer to Figure 6-11 for a flow chart of this instruction.

INPW
Word-Addressed
Input to Storage

P



P + 1



B = "1" for backward storage
ch = I/O channel designator, 0-7
INT = "1" for interrupt upon completion
N = "0" for 12- to 24-bit assembly
N = "1" for no assembly
m = first word address of I/O data block; becomes current address as I/O operation progresses
n = last word address of input data block, plus one (minus one, for backward storage)
G = "1" for word count control
Bits 15 and 16 at P and P + 1 should be loaded with zeros.

Instruction Description: This instruction transfers 12-bit bytes or 24-bit words from an external equipment to storage. If channel 'ch' is not busy the buffered input operation commences while Main Control performs a RNI at P + 3. If channel 'ch' is initially busy, Main Control performs a RNI at P + 2 and the input operation does not occur.

Comments: During 12- to 24-bit assembly with forward storage specified, the first byte of data is stored in the upper half of the first word specified by address 'n'. For backward storage, the first byte is stored in the lower half.

During no assembly, data is stored as received from the peripheral. When using a 12-bit channel, bytes are always stored in the lower half of storage words; the upper half remains unchanged. If a 24-bit channel is used with a 12-bit device, the upper half of storage words are filled with zeros.

NOTE

When bit 20 of the sub-instruction word at P + 1 is a "1", the word count control feature allows the input operation to continue beyond an End-of-Record signal(s).

Refer to Figure 6-11 for a flow chart of this instruction.

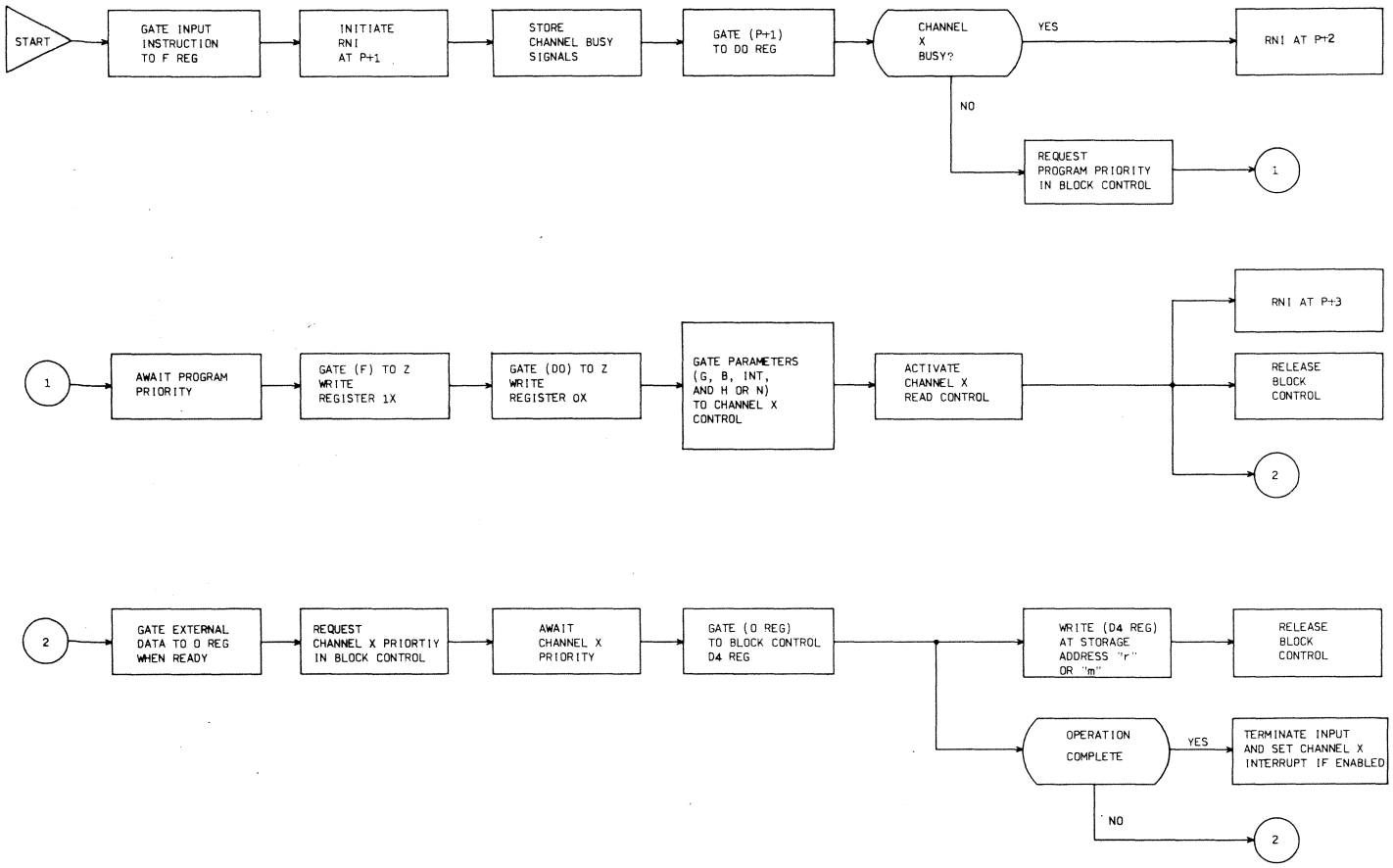
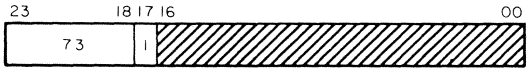


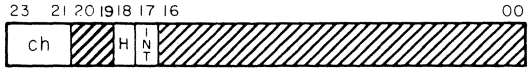
Figure 6-11. Input to Storage (73 and 74) Operation

INAC
Input Character to A

P



P+1



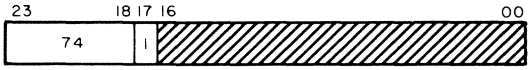
ch = I/O channel designator, 0-7
H = "0" for 6-bit transfer
H = "1" for 12-bit transfer
INT = "1" for interrupt upon completion
Bits 00 - 16 of P and P + 1, and bits 19 and 20 of P + 1 should be loaded with zeros.

Instruction Description: This instruction transfers either a 6-bit character into the lower 6 bits of A, or a 12-bit byte into the lower 12 bits of A from an external equipment. The (A) are cleared prior to loading. When the input to A is completed, a RNI is made at P + 3. If channel 'ch' is initially busy, a RNI is performed at P + 2.

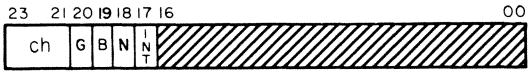
Refer to Figure 6-12 for a flow chart of this instruction.

INAW
Input Word to A

P



P+1



B = "1" for backward
ch = I/O channel designator, 0-7
N = "0" for 12- to 24-bit assembly
N = "1" for no assembly
INT = "1" for interrupt upon completion
G = "1" for word count control
Bits 00 - 16 of P and P + 1 should be loaded with zeros.

Instruction Description: This instruction transfers either a 12-bit byte or a 24-bit word from an external equipment to the A register. The (A) are cleared prior to the input. When the input to A is completed, a RNI is made at P + 3. If channel 'ch' is initially busy, a RNI is performed at P + 2.

Comments: If no assembly is specified using an equipment with a 12-bit interface, a byte is transferred into the lower 12 bits of A. A 24-bit word is transferred to A if 24-bit channel/equipment is used.

When using the channel's assembly feature in 12-bit mode, the first of the two bytes is loaded into the upper half of A during forward assembly, or into the lower half when backward assembly is specified.

NOTE

When bit 20 of the sub-instruction word at P + 1 is a "1", the word count control feature allows the input operation to continue beyond an End-of-Record signal.

Refer to Figure 6-12 for a flow chart of this instruction.

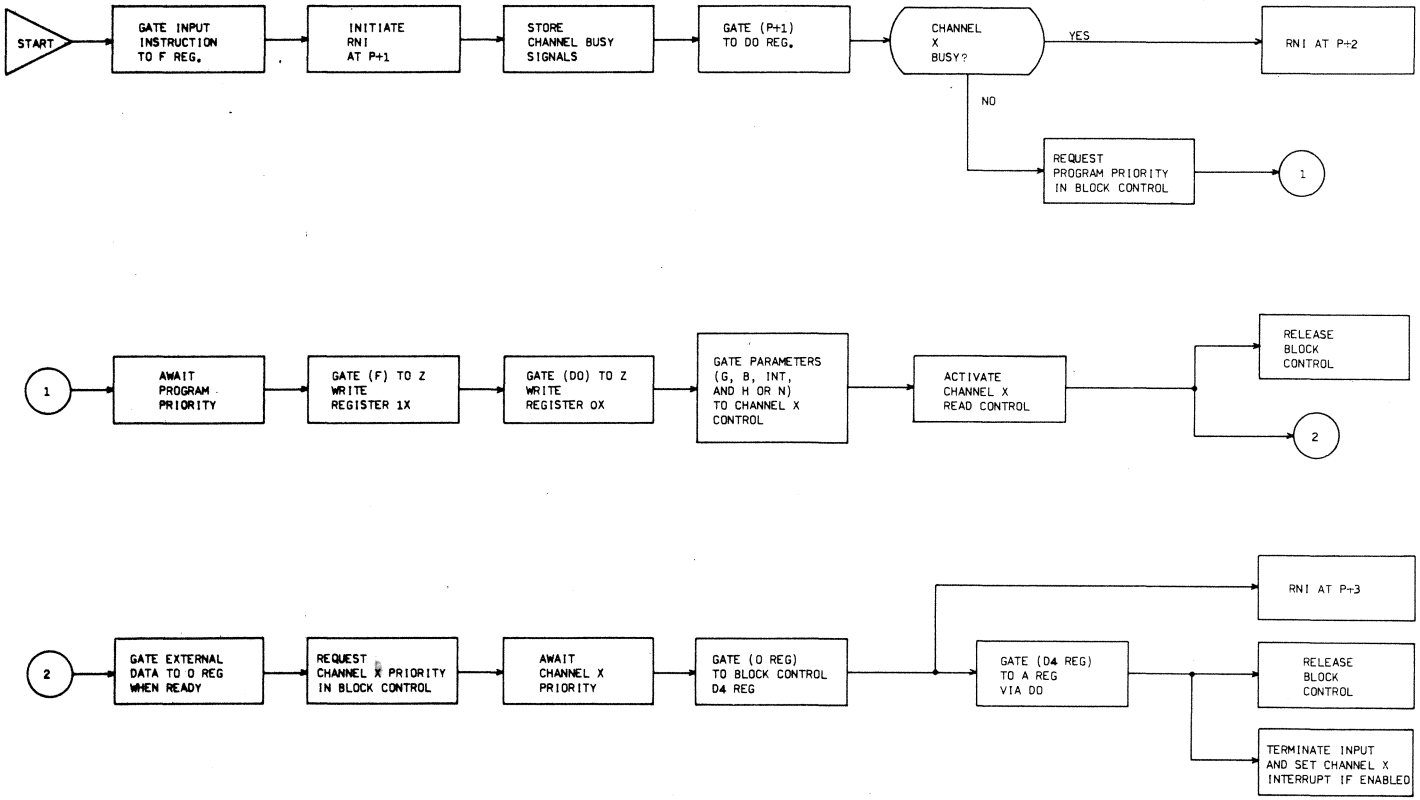
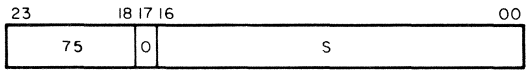


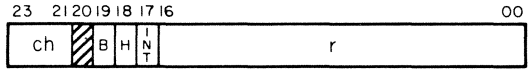
Figure 6-12. Input to A (73 and 74) Operation

OUTC
Character-Addressed Output from Storage

P



P+1



B = "1" for backward storage
ch = I/O channel designator, 0-7
H = "0" for 24- to 6-bit disassembly
H = "1" for 24- to 12-bit disassembly
INT = "1" for interrupt upon completion
r = first character address of I/O data block; becomes current address as I/O operation progresses
s = last character address of output data block, plus one (minus one, for backward output)
Bit 20 at P + 1 should be loaded with a "0"

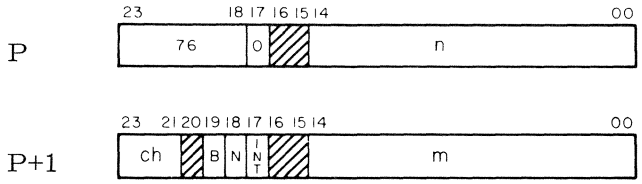
Instruction Description: This instruction transfer 6-bit characters or 12-bit bytes from storage to an external equipment. If channel 'ch' is not initially busy, the buffered output operation from storage commences while Main Control simultaneously performs a RNI at P + 3. If channel 'ch' is initially busy, Main Control performs a RNI at P + 2 and the output operation does not occur.

Comments: During 24- to 6-bit disassembly, 6-bit characters are removed from storage, beginning with character address 'r'. The next character is transferred from 'r + 1' if doing forward storage, or from address 'r - 1' if backward storage is specified.

During 24- to 12-bit disassembly, successive 12-bit bytes are transferred from successive halves of storage words. The lowest bit of both character addresses is forced to "0" for the operation, insuring that 12-bit bytes are disassembled from either the upper or lower half of a word. To insure transfer of a correct number of characters, an even character count must be used.

Refer to Figure 6-13 for a flow chart of this instruction.

OUTW
Word-Addressed
Output from Storage



B = "1" for backward storage
ch = I/O channel designator, 0-7
INT = "1" for interrupt upon completion
m = first word address of I/O data block; becomes current address as I/O operation progresses
N = "0" for 24- to 12-bit disassembly
N = "1" for straight 12- or 24-bit data transfer
n = last word address of output data block, plus one (minus one, for backward output)

Bits 15 and 16 at P and bits 15, 16, and 20 of P + 1 should be loaded with zeros.

Instruction Description: This instruction transfers 12-bit bytes or 24-bit words from storage to an external equipment. If channel 'ch' is not initially busy, the buffered output operation from storage commences while Main Control simultaneously performs a RNI at P + 3. If channel 'ch' is initially busy, Main Control performs a RNI at P + 2 and the output operation does not occur.

Comments: During 24- to 12-bit disassembly with forward storage specified, the first byte of data is removed from the upper half of the word specified by word address 'n'. For backward storage, the first byte is removed from the lower half.

When using an equipment with a 12-bit interface to the channel and specifying straight transfer (N=1), only the lower 12 bits of consecutive storage locations are transferred. The upper 12 bits are lost.

Refer to Figure 6-13 for a flow chart of this instruction.

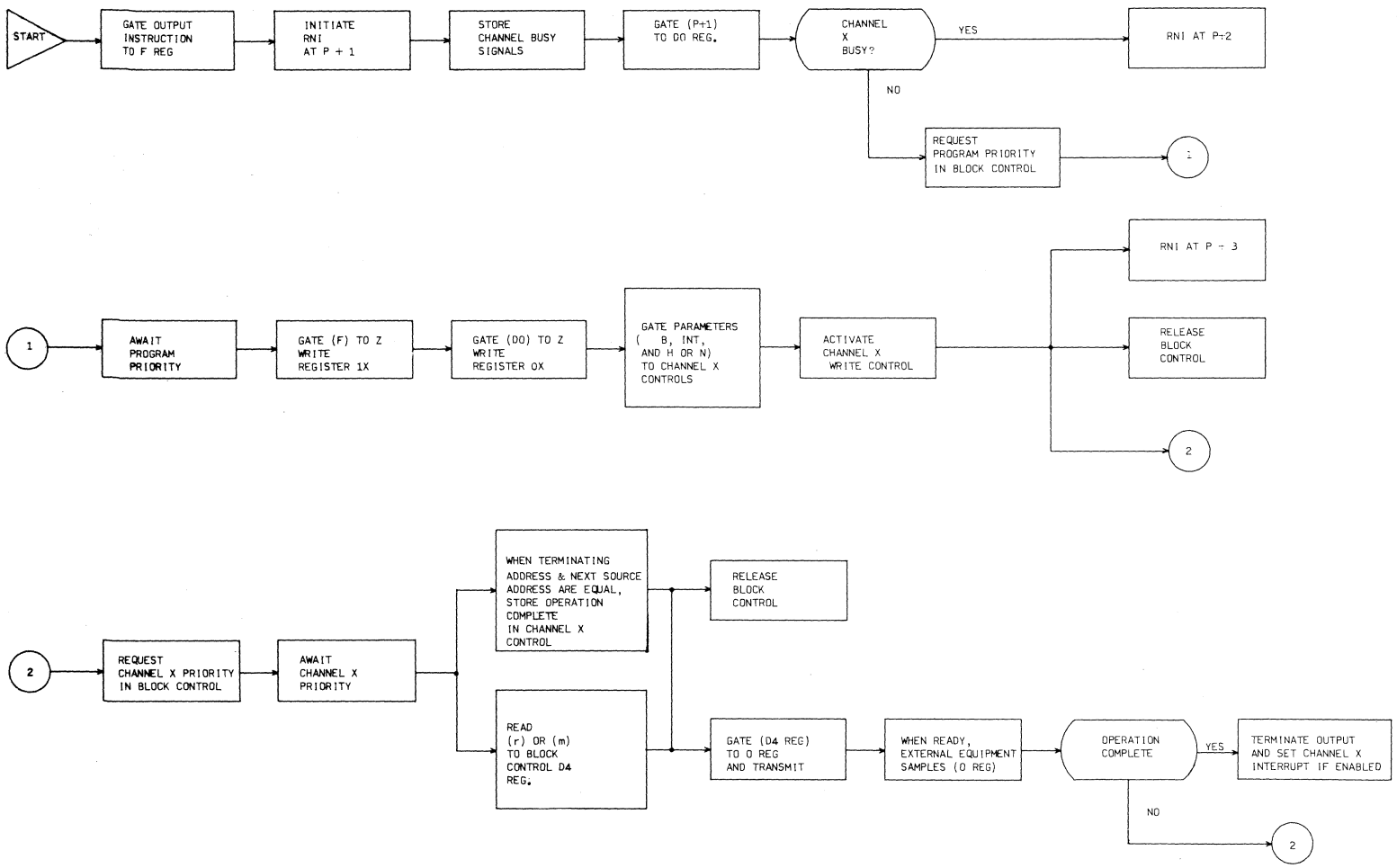
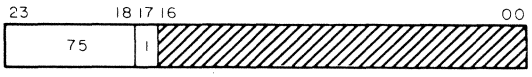


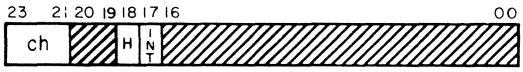
Figure 6-13. Output from Storage (75 and 76) Operation

OTAC
Output
Character from A

P



P+1



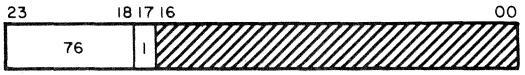
ch = I/O channel designator, 0-7
H = "0" for 6-bit output
H = "1" for 12-bit output
INT = "1" for interrupt upon completion
Bits 00 - 16 of P and P + 1, and bits 19 and 20 of P + 1 should be loaded with zeros.

Instruction Description: This instruction transfers the lower 6 bits of A or the lower 12 bits of A to an external equipment. The original (A) remain unchanged. Main Control performs a RNI at P + 3 as soon as the (A) are transferred to the channel. The channel continues to transmit the data to the peripheral until it is accepted. If channel 'ch' is initially busy, the operation cannot be performed and a RNI is made from P + 2.

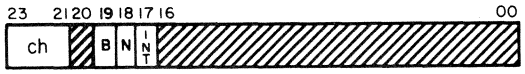
Refer to Figure 6-14 for a flow chart of this instruction.

OTAW
Output Word from A

P



P+1



B = "1" for backward
N = "0" for 24- to 12-bit disassembly
N = "1" for straight 12- or 24-bit output
ch = I/O channel designator, 0-7
INT = "1" for interrupt upon completion
Bits 00 - 16 of P and bits 00 - 16 and bit 20 of P + 1 should be loaded with zeros.

Instruction Description: This instruction transfers either the lower 12 bits of A or all 24-bits of A to an external equipment. The original (A) remain unchanged. Main Control performs a RNI at P + 3 as soon as the (A) are transferred to the channel. The channel continues to transmit the data to the peripheral until it is accepted. If channel 'ch' is initially busy, the operation cannot be performed and a RNI is made from P + 2.

Comments: If straight transfer is specified (N=1) when using an equipment with a 12-bit interface, the output is from the lower 12 bits of A. All 24 bits of A are transferred when using a 24-bit channel and equipment.

When using the channel's disassembly feature in 12-bit mode, the first of the two bytes is taken from the upper half of (A) during forward disassembly, or from the lower half of (A) when backward disassembly is specified.

Refer to Figure 6-14 for a flow chart of this instruction.

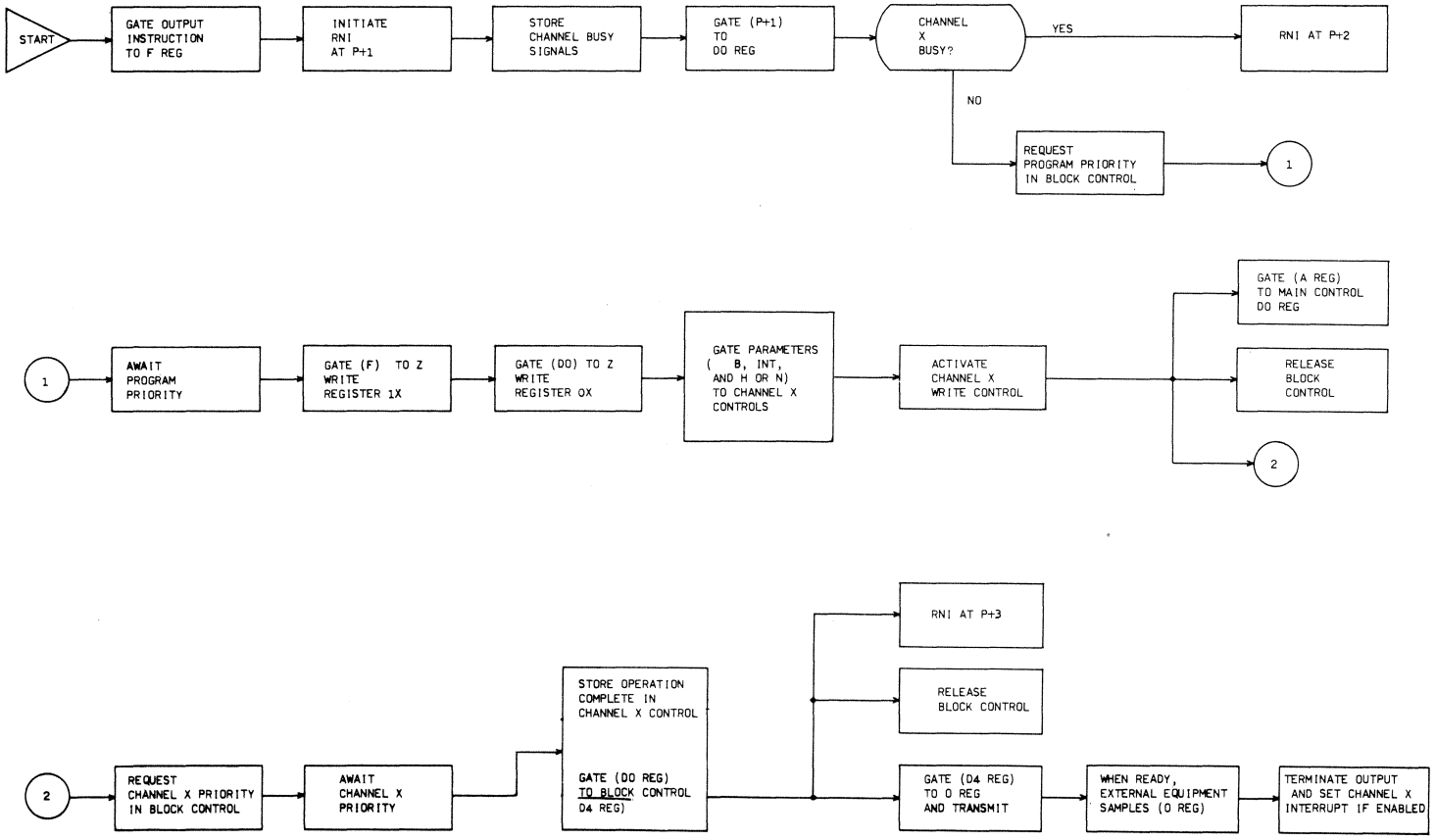


Figure 6-14. Output from A (75 and 76) Operation

Business Data Processing Instructions

The BDP in 3514 Central Processor models is somewhat different from the BDP in the 3504 Central Processor. The main differences are:

1. The 3504 BDP has instructions for BCD to ASCII conversion (66.2) and ASCII to BCD conversion. These instructions are not available in the 3514 BDP.
2. The Compare instructions (67.3) are quite different in the two BDP's.
3. The 3514 BDP Move instruction (64.0) allows full word move operations as well as character move operations. In the 3504 the 64.0 instruction is limited to character move operations.
4. There are minor differences in several other instructions.

BDP Instruction Set

Operation Field		Address Field	Interpretation
MVE ***	64.0	r, B_r, $1, s, B_s, $2	Move field A to field C
MVE, dc ***	64.0	r, B_r, s, B_s, $2	Move field A to field C with delimiting
MVBF	64.1	r, B_r, $1, s, B_s, $2	Move field A to field C and blank fill
MVZF	64.2	r, B_r, $1, s, B_s, $2	Move field A to field C and zero fill
MVZS	64.3	r, B_r, $1, s, B_s, $2	Move field A to field C and suppress zeros
MVZS, dc	64.3	r, B_r, s, B_s, $2	Move field A to field C and suppress zeros with delimiting
ZADM***	67.2	r, B_r, $1, s, B_s, $2	Move field A to field C and add zeros
FRMT	64.4	r, B_r, $1, s, B_s, $2	Move field A to field C and format with commas and decimal point
EDIT	64.4	r, B_r, $1, s, B_s, $2	Move field A to field C and perform complete COBOL edit
SCAN, LR, EQ	65.0	r, B_r, $2, sc	Scan field (left to right) for equal condition
SCAN, LR, EQ, dc	65.0	r, B_r, $2, sc	Scan field (left to right) for equal condition with delimiting
SCAN, LR, NE	65.2	r, B_r, $2, sc	Scan field (left to right) for unequal condition
SCAN, LR, NE, dc	65.2	r, B_r, $2, sc	Scan field (left to right) for unequal condition with delimiting

*3504 BDP Only
**3514 BDP Only
***Minor differences between 3504 and 3514. See instruction descriptions.

SCAN, RL, EQ	65.1	r, B_r, $2, sc	Scan field (right to left) for equal condition
SCAN, RL, EQ, dc	65.1	r, B_r, $2, sc	Scan field (right to left) for equal condition with delimiting
SCAN, RL, NE	65.3	r, B_r, $2, sc	Scan field (right to left) for unequal condition
SCAN, RL, NE, dc	65.3	r, B_r, $2, sc	Scan field (right to left) for unequal condition with delimiting
CVDB	66.0	r, B_r, $1, m, B_m	Convert BCD field to binary field
CVBD	66.1	m, B_m, n, B_n	Convert binary field to BCD field
DTA *	66.2	r, B_r, $2, m, B_m	Convert BCD field to ASCII field
DTA, dc *	66.2	r, B_r, $2, m, B_m	Convert BCD field to ASCII delimiting
ATD *	66.3	m, B_m, $2, s, B_s	Convert ASCII field to BCD field
ATD, dc *	66.3	m, B_m, $2, s, B_s	Convert ASCII field to BCD, delimiting
PAK	66.4	r, B_r, $2, m, B_m	Convert 6-bit BCD to 4-bit BCD
UPAK	66.5	m, B_m, s, B_s, $2	Convert 4-bit BCD to 6-bit BCD
ADM ***	67.0	r, B_r, $1, s, B_s, $2	Add field A to field C
SBM ***	67.1	r, B_r, $1, s, B_s, $2	Subtract field A from field C
CMP *	67.3	r, B_r, $1, s, B_s, $2	Compare field A to field C
CMP, dc *	67.3	r, B_r, s, B_s, $1	Compare field A to field C delimiting
CMP **	67.3	r, B_r, $1, s, B_s, $2	Collating compare of field A with field C
CMP, n **	67.3	r, B_r, $1, s, B_s, $2	Numeric compare of field A with field C
TST	67.4	r, B_r, $1	Test field A for sign
TSTN	67.4	r, B_r, $1	Test field A for numeric
LBR ***	70.6	m	Load BDP
SBR	70.7	m	Store BDP

*3504 BDP Only
**3514 BDP Only
***Minor differences between 3504 and 3514. See instruction descriptions.

NOTE

All instructions in this group (except LBR and SBR) are unconditionally trapped when the BDP MODE switch is OFF or the optional BDP is not present. LBR and SBR are also trapped if the switch is OFF during Non-Executive mode or Program state of Executive mode. However, during Monitor state, they are No-Ops.

Whenever one of the 64 - 70 instructions is read from memory during execution of a program, the Main Control section signals the BDP section of the Central Processor to assume control for the instruction. Main Control performs all required index and memory operations. Generally, the BDP instructions involve operations with variable length data fields and certain guidelines should be followed while programming.

In those instructions using two variable length data fields, care must be taken in assigning these fields to memory so that overlapping of processed data of the result field over unprocessed data of the source field does not occur. If overlapping occurs the results will be unpredictable.

Interrupts During BDP Instructions

Interrupts are recognized near the end of the first RNI of all instructions. However, after the first RNI of BDP instructions, Main Control continually tests for active interrupt conditions. If a selected interrupt (or Abnormal interrupt) condition becomes active, an Interrupt Stop signal is sent to the BDP section. The BDP relinquishes control after the current character operation is completed. The interrupt is actually recognized as Main Control rereads the instruction at P, or at the address of the next instruction if the current instruction was completed.

The BDP records interrupt recovery conditions (refer to the LBR instruction), and enables operating information to the B^3 register. If recovery from interrupts is desired, the interrupt routine used must contain a SBR instruction to store the recorded interrupt recovery conditions, and a LBR instruction to return the recovery conditions to the BDP once the interrupt processing is completed. These conditions normally enable a restart to be made from the point of interrupt. Exceptions to the recovery start are: the 66.0 and 66.1 instructions always restart from the beginning if interrupted, and if the interrupt is because of an Illegal Write, the instructions 66.4 and 66.5 also restart from the beginning.

The (B^3) register has the following significance when a BDP instruction is interrupted:

Bits 00 - 11, record the count of the Field C characters processed prior to interrupt.

Bit 12 = "1", if a second pass (complementing operation) was in progress.

Bit 13 = "1", if an arithmetic carry was generated on an ADM or SBM instruction during the iteration preceeding interrupt. This is an internal status bit used to enable interrupt recovery and does not indicate an Arithmetic Overflow at instruction completion.

Bit 14 = "1", if a BCD fault occurred.

BDP Condition Register

The BDP Condition register (BCR) is a 2-bit register that is set to indicate conditions existing directly after a business data processing instruction has been executed. The BCR is cleared upon execution of the next BDP instruction. The (BCR) can be sampled to condition jumps to address 'm' by the three jump instructions: JMP, ZRO; JMP, HI; JMP, LOW. Refer to the Jump Instructions group earlier in Section 6 for these instructions and the BCR codes.

Numeric Fields

Six-bit numeric BCD characters consist of a numeric portion (lower 4-bits) and a zone portion (upper 2 bits), the latter of which specifies sign for the character. When considering variable-length numeric fields, the convention followed is to designate the field sign with the sign of the lowest order (rightmost) character in the field. This lowest order character is hereafter referred to as the sign character. The zone bits for all other characters in the field must equal zero.

The sign of fields in packed BCD (4-bit) is specified by two special 4-bit sign characters (1010_2 -positive, and 1011_2 - negative) in the lowest order character position.

The significance of zone bits and the numeric portion of 6-bit BCD characters are shown below:

Sign of BCD Field	Relative Bit Positions	
	6	5
+	0	0
+	0	1
-	1	0
+	1	1

Decimal Number	BCD Character Relative Bit Positions			
	4	3	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

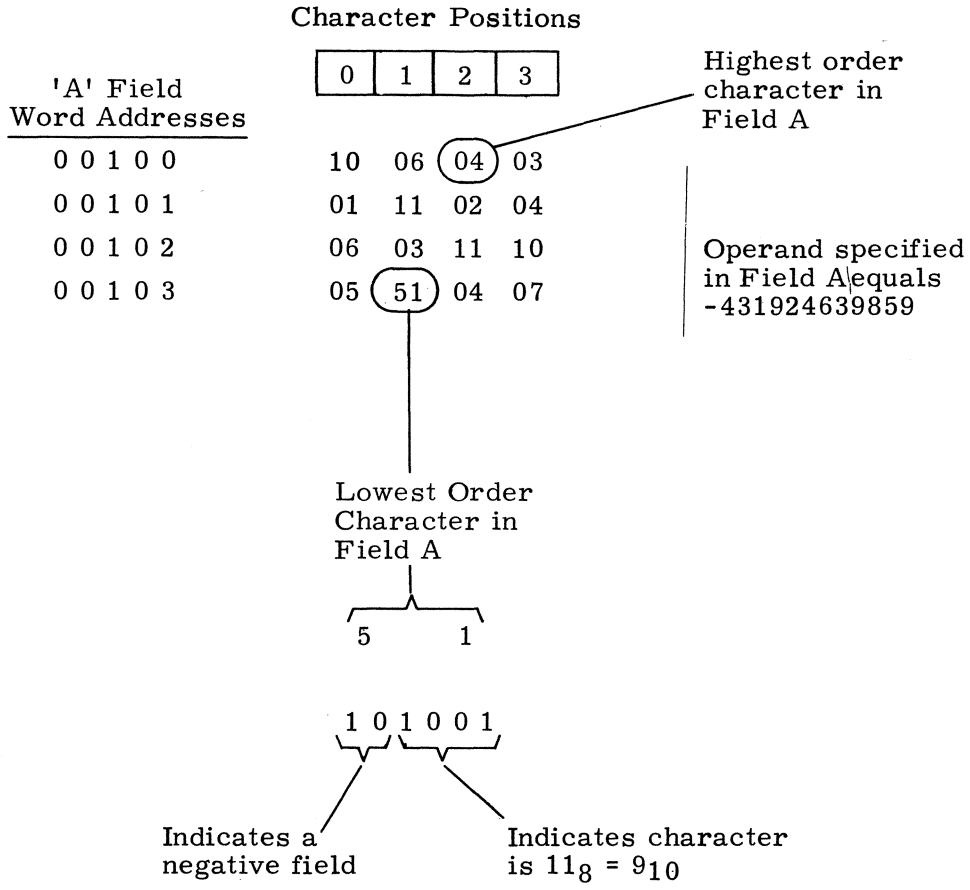
EXAMPLE: Following is an example illustrating execution of a MVZF instruction:

P = 64000202
P + 1 = 27003000 (B^1) = 00200
P + 2 = 00140017 (B^2) = 01000

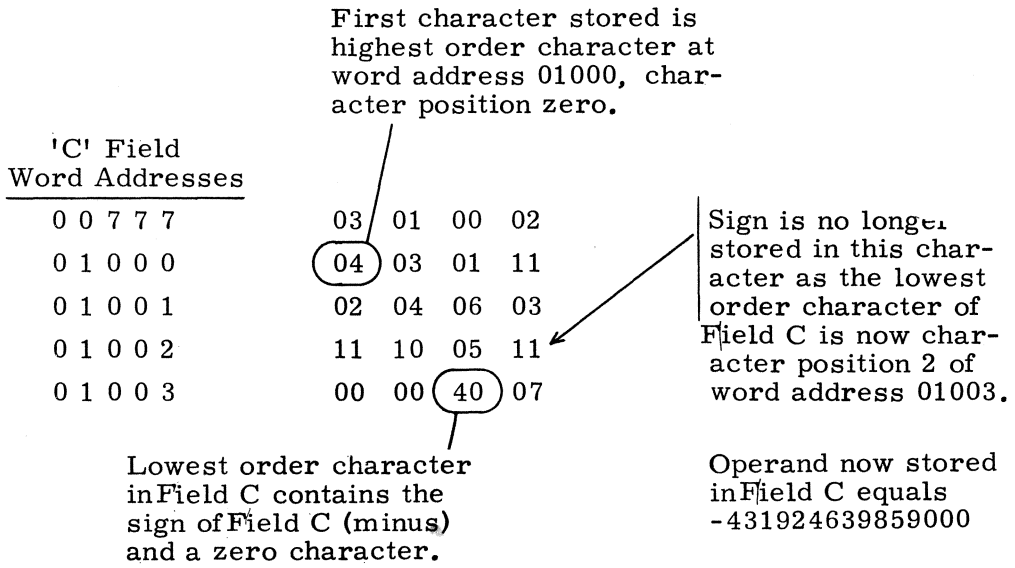
Analysis:

1. The unmodified character address 'r' is 00202.
2. B_r = 3, requiring (B^1) be added to r. If (B^1) = 00200 then R = 00402 which equals word address 00100 character position 2. This is the true address of the highest order character in field A.
3. B_s = 2, requiring (B^2) be added to the unmodified character address 's', 03000. If (B^2) = 01000, then S = 04000.
4. The length of the A field is 14_8 characters and the alloted length of the C field is 17_8 characters. The last three characters of field C will be filled with zeros. The last character of field C (a zero) will also contain the sign of the field.

(continued on next page)

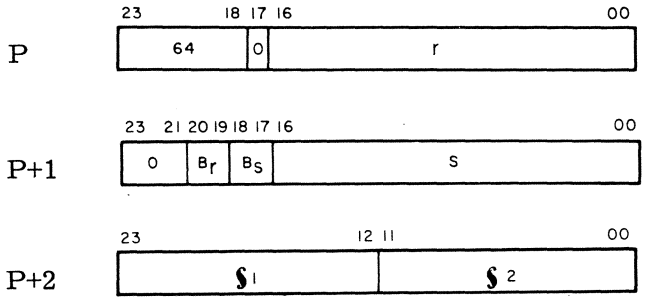


The operation proceeds as follows:



MVE
Move Field A to Field C

Applicable to 3504 BDP only. See next page for 3514 BDP.



r = unmodified address of the highest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the highest order character in field C.
S = s + [B_s]

B_s = index register flag for field C (same bit functions as B_r)

$\$_1$ = number of characters in field A to be moved

$\$_2$ = number of available character positions in field C

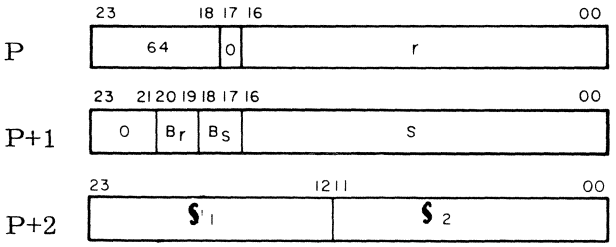
Instruction Description: Move a field of up to 4095 6-bit alphanumeric characters from field A to field C, left to right. If field lengths are unequal, the length of the shorter field terminates the move and the remainder of the longer field is not moved or changed.

Comments: The BDP Condition register is set to the sign of field A if the sign character is moved. It is set to 00_2 for a positive sign (or no sign transferred) and to 10_2 for a negative sign.

Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

MVE
Move Field A to Field C

Applicable to 3514 BDP only. See previous page for 3504 BDP.



r = unmodified address of the highest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the highest order character in field C.
S = s + [B_s]

B_s = index register flag for field C (same bit functions as B_r)

$\$_1$ = number of characters in field A to be moved

$\$_2$ = number of available character positions in field C

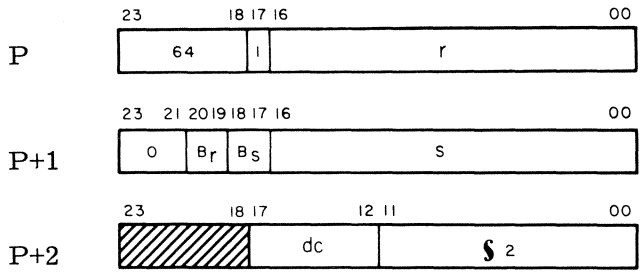
Instruction Description: Move a field of up to 4095 6-bit alphanumeric characters from field A to field C, left to right. If field lengths are unequal, the length of the shorter field terminates the move and the remainder of the longer field is not moved or changed.

Comments: To increase performance of the move, object data should be organized on word boundaries. A word move is performed if starting addresses (after indexing) define character positions that are the same in both fields and the number of characters to be moved is greater than 4.

Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

The BDP Condition register is not used for this instruction.

MVE, dc
Move Field A to Field C, Delimited



r = unmodified address of the highest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the highest order character in field C.
S = s + [B_s]

B_s = index register flag for field C (same bit functions as B_r)

$\$_2$ = number of available character positions in both field A and field C

dc = 6-bit delimiting character compared against the characters in field A

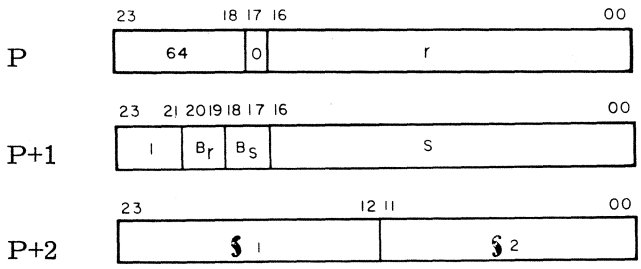
Bits 18 - 23 of P + 2 should be loaded with zeros.

Instruction Description: Move a field of up to 4095 6-bit alphanumeric characters from field A to field C, left to right. The length of field C, $\$_2$, terminates the move operation. If the delimiting character is recognized at any time during the character move, the operation is terminated after the delimit character has been moved.

Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

The BDP Condition register is not used for this instruction.

MVBF
Move Field A to Field C and Blank Fill



r = unmodified address of the highest order character in field A.
$R = r + [B_r]$

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the highest order character in field C.
$S = s + [B_s]$

B_s = index register flag for field C (same bit functions as B_r)

$§_1$ = number of characters in field A to be moved

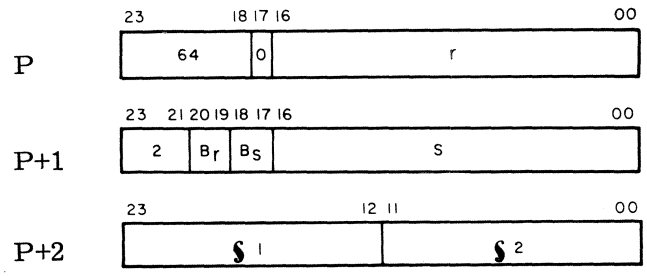
$§_2$ = number of available character positions in field C

Instruction Description: Move a field of up to 4095 6-bit alphanumeric characters from field A to field C, left to right. If field lengths are unequal, the length of the shorter field terminates the move. If field C is longer, its remainder is filled with blanks.

Comments: The BDP Condition register is set to the sign of field A if the sign character is moved. It is set to 00_2 for a positive sign (or no sign transferred) and to 10_2 for a negative sign.

Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

MVZF
Move Field A to Field C and Zero Fill



r = unmodified address of the highest order character in field A.
$R = r + [B_r]$

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the highest order character in field C.
$S = s + [B_s]$

B_s = index register flag for field C (same bit functions as B_r)

$§_1$ = number of characters in field A to be moved

$§_2$ = number of available character positions in field C

Instruction Description: Move a field of up to 4095 6-bit BCD numeric characters from field A to field C, left to right. If field lengths are unequal, the shorter field terminates the move. If field C is longer, its remainder is filled with zeros.

Comments: The zone bits of all field A characters are forced to zero when moved to field C, as is any field A character containing a 12_8 - 17_8 code.

The sign from field A is always transferred to the BDP Condition register and to the lowest order character in field C (which may be a zero-filled character), even if $§_1 > §_2$.

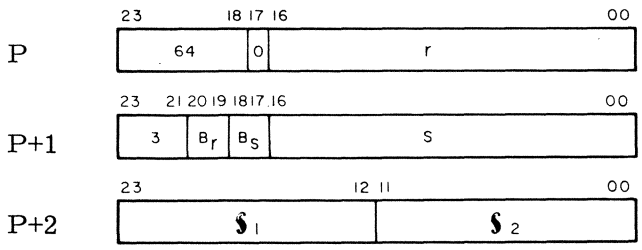
Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

A BCD fault is generated if one of the following conditions occur:

1. Zone portion of any character in field A (except sign character) does not equal zero.
2. Numeric portion of any character in field A contains a BCD code greater than 11_8, except the sign character where a 12_8 code is legal.
3. The sign character contains a 72_8 code.

Operation continues despite any BCD fault.

MVZS
Move Field A to Field C and Zero Suppress



r = unmodified address of the highest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the highest order character in field C.
S = s + [B_S]

B_S = index register flag for field C (same bit functions as B_r)

$\$_1$ = number of characters in field A to be moved

$\$_2$ = number of available character positions in field C

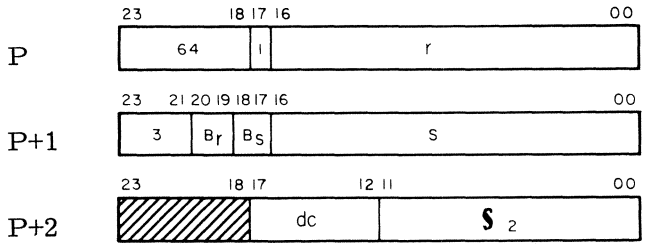
Instruction Description: Move a field of up to 4095 6-bit alphanumeric characters in field A to field C, left to right, and replace all leading zeros occurring in field A with blanks in field C. If field lengths are unequal, the shorter field terminates the move and the remainder of the longer field is not moved or changed.

Comments: If field A is longer than field C, the sign bits may be invalid since sign bits are not checked or modified and consist of the zone bits of the last character moved.

The BDP Condition register is not used (always set to 00_2).

Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

MVZS, dc
Move Field A to Field C and Zero Suppress, Delimited



r = unmodified address of the highest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the highest order character in field C.
S = s + [B_s]

B_s = index register flag for field C (same bit functions as B_r)

$\$_2$ = number of available character positions in both field A and field C

dc = 6-bit delimiting character compared against the characters in field A

Bits 18 - 23 of P + 2 should be loaded with zeros.

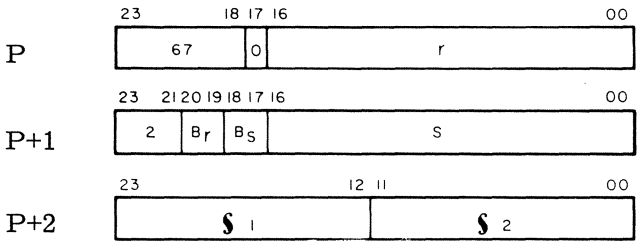
Instruction Description: Move a field of up to 4095 6-bit alphanumeric characters from field A to field C, left to right, and replace all leading zeros occurring in field A with blanks in field C. The length of field C, $\$_2$, terminates the move operation. If the delimiting character is recognized at any time during the character move, the operation is terminated after the delimit character has been moved.

Comments: The sign of the newly formed C field may be invalid since sign bits are not checked or modified and consist of the zone bits of the last character moved .

The BDP Condition register is always set to a positive sign (00_2).

ZADM
Zero and Add Move

Applicable to 3504 BDP only. See next page for 3514 BDP.



r = unmodified address of the lowest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the lowest order character in field C.
S = s + [B_s]

B_s = index register flag for field C (same bit functions as B_r)

§1 = number of characters in field A to be moved

§2 = number of available character positions in field C

Instruction Description: Move a field of up to 4095 6-bit BCD numeric characters from field A to field C, right to left. If field lengths are unequal, the shorter field terminates the move. If field C is longer, its remainder is filled with zeros.

Comments: The algebraic sign of field A is obtained from the zone bits of the lowest order character in field A (10_2 indicates a negative field, and all other combinations indicate positive field). This sign is stored in the sign character in field C. The BDP Condition register is also set to the sign of field A.

Any field A character with a 12_8 - 17_8 code is forced to zero when moved.

Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

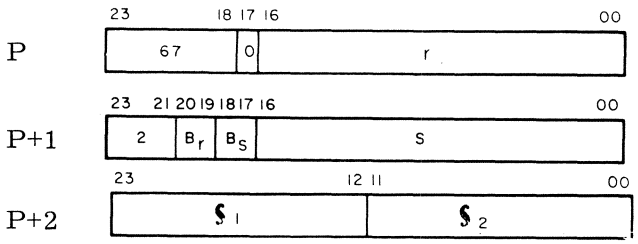
A BCD fault is generated if one of the following conditions occur:

1. Zone portion of any character in field A (except sign character) does not equal zero.
2. Numeric portion of any character in field A contains a BCD code greater than 11_8, except the sign character where a 12_8 code is legal.
3. The sign character contains a 72_8 code.

Operation continues despite any BCD fault.

ZADM
Zero and Add Move

Applicable to 3514 BDP only. See previous page for 3504 BDP.



r = unmodified address of the lowest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the lowest order character in field C
S = s + [B_S]

B_S = index register flag for field C (same bit functions as B_r)

$\$_1$ = number of characters in field A to be moved

$\$_2$ = number of available character positions in field C

Instruction Description: Move a field of up to 4095 6-bit BCD numeric characters from field A to field C, right to left. If field lengths are unequal, the shorter field terminates the move. If field C is longer, the remainder of field C is filled with zeros.

Comments: The algebraic sign of field A is obtained from the zone bits of the lowest order character in field A. (10_2 indicates a negative field, all other combinations indicate a positive field.) This sign is stored in the sign character of field C. The BDP Condition register is also set to the sign of field A.

Any field A character with a 12_8 - 17_8 code is forced to zero when moved. A sign character of 40_8 in the result field is converted to 52_8, the code for magnetic tape character negative zero. A 60_8 code encountered in either field is treated as zero.

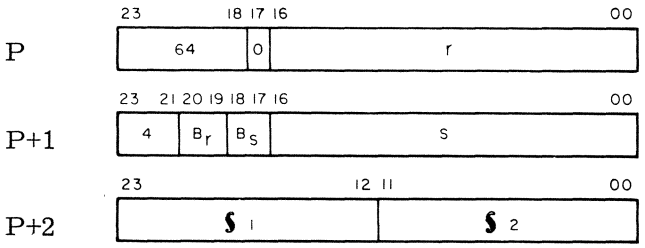
Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

A BCD fault is generated if one of the following conditions occur:

1. Zone portion of any character in field A (except sign character) does not equal zero.
2. Numeric portion of any character in field A contains a BCD code greater than 11_8, except the sign character where a 12_8 code is legal.
3. The sign character contains a 72_8 code.

Operation continues despite any BCD fault.

FRMT
Move Field A and Format in Field C



r = unmodified address of the highest order character in field A.
$R = r + [B_r]$

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the highest order character in field C.
$S = s + [B_S]$

B_S = index register flag for field C (same bit functions as B_r)

$\$_1$ = number of characters in field A to be edited. (Values must be 2, 5, 10_8, 13_8, 16_8, etc.)

$\$_2$ = number of characters in field C. (Values of $\$_2$ must be 3, 6, 12_8, 16_8, 22_8, etc. and include decimal point and commas.)

Instruction Description: Move the numeric characters in field A from left to right into field C, replacing leading zeros with blanks and inserting a comma after every three characters moved. A decimal point is inserted in the third lowest order position of the C field.

Comments: Leading zeros in field A, together with normally inserted commas, are suppressed until the first non-zero character is encountered. Zero suppression terminates with the decimal point if only zeros are encountered. The sign of field A is recorded in the BDP Condition register, but does not appear in the resultant C field.

Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

A BCD fault is generated if one of the following conditions occur:

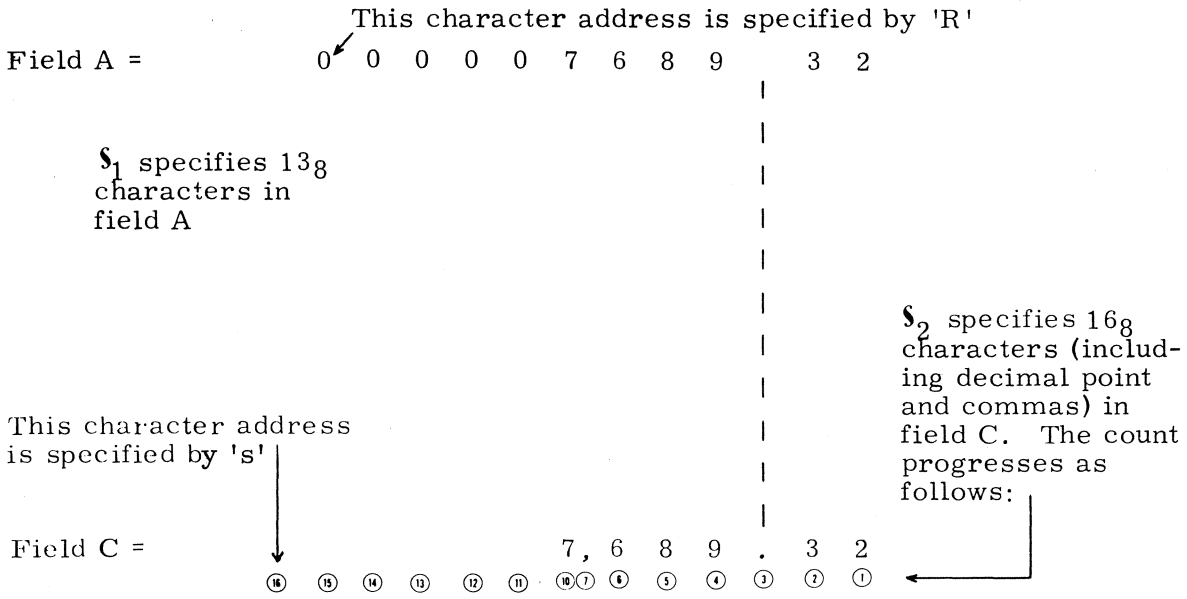
1. Zone portion of any character in field A (except sign character) does not equal zero.
2. Numeric portion of any character in field A contains a BCD code greater than 11_8 except the sign character, where a 12_8 code is legal.
3. The sign character contains a 72_8 code.
4. Incorrect field length specified: a) the two fields are not aligned, e.g., $\$_1$ = 2 and $\$_2$ = 6, b) illegal count used, e.g., $\$_2 \neq 3, 6, 12_8, 16_8, 22_8$, etc.

Operation continues despite any BCD fault.

EXAMPLE:

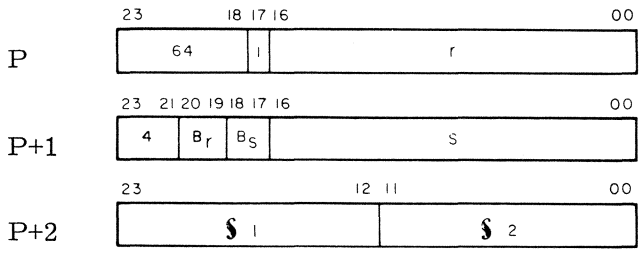
$\$_1 = 13_8$

$\$_2 = 16_8$



The leading zeros are removed, leaving six blanks in field C.

EDIT
Move Field A and Edit in Field C



r = unmodified address of the highest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the highest order character in field C.
S = s + [B_S]

B_S = index register flag for field C (same bit functions as B_r)

$1 = number of characters in field A to be edited

$2 = number of characters in field C

Instruction Description: Perform an edit on the numeric characters in field A proceeding from left to right. The editing is performed character for character with respect to the COBOL type editing characters in field C. The resulting edited field is stored in field C.

Comments: Programming consideration must be given to aligning the characters in field A with the proper editing characters in field C. The COBOL type editing characters used in field C and applicable to this instruction are listed here with their descriptions. Any other character in the C field is recognized as a 9.

The BDP Condition register is set to the sign of field A. The conditions for a BCD Fault are the same as for FRMT, except condition 4 does not apply.

Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

Editing examples are listed following the character descriptions:

EDITING CHARACTERS

9 $ + - . , 0 B CR DB Z * /

- Direct Characters:

  9 When the character 9 appears in the C field, it is replaced by the corresponding character in the A field.

Insertion Characters:

When an insertion character is specified in the C field, it remains in that character position in the edited field. The insertion characters are:

$ + - . , 0 B CR DB /

$ When a single dollar sign is specified as the left-most symbol in a C field, it appears as the left-most character in the edited field. This character is included in the character count ($\S_2$) of the edited field.

+ When a plus sign is specified as the first or last symbol of a C field, a plus sign is inserted in the indicated character position of the edited data, provided the field of data is positive or is unsigned. If the data is negative, a minus is inserted in the indicated character position. The sign is included in the character count ($\S_2$) of the edited field.

- When a minus sign is specified as the first or last symbol of a C field, a minus sign is inserted in the indicated character position of the edited data, provided the field of data is negative. If the data is not negative, a blank is inserted in the indicated character position. The sign or blank is included in the character count ($\S_2$) of the edited field.

. (decimal point) This character is used in a C field to represent an actual decimal point as opposed to an assumed decimal point. When used, a decimal point appears in the edited data as a character in the same character position as it appears in the C field and it is included in the character count of the edited field. A picture of a report item can never contain more than one decimal point, actual or assumed.

, When a comma is used in a C field, a comma is inserted in the corresponding character position of the edited data. It is included in the character count ($\S_2$) of the edited field.

0 (zero) When a zero is used in a C field, a zero is inserted in the corresponding character position in the edited field. It is included in the character count ($\S_2$) of the edited field.

B When the character, B, is used in a C field, a blank is inserted in the corresponding character position in the edited field. It is included in the character count ($\S_2$) of the edited field.

CR The combined characters (CR) represent a credit in accounting operations and may be specified only at the right end of a C field. The symbol is inserted in the last two character positions of the edited field, provided the value of the data is negative. If the data is positive or unsigned these last two character positions are set to blanks. Since this symbol always results in two characters (CR or blanks) it is included as two characters in the character count ($\S_2$) of the edited field.

DB The combined characters (DB) represent a debit and may be specified only at the right end of a C field. It has the same results as the credit symbol, using DB or blanks.

(slash) When a slash (/) is used in a C field, a slash is inserted in the corresponding character position in the edited field. It is included in the character count ($\S_2$) of the edited field.

- Replacement Characters:

A replacement character in the C field suppresses leading zeros in data and replaces them with other characters in the edited data. The replacement characters are:

Z * $ + -

Only one type of replacement character may be used in a picture, although a Z or asterisk (*) may be used with any of the insertion characters including:

$ + -

Z One Z character is specified at the left end of a C field for each leading zero in the A field that requires suppression and replacement by a blank in the edited field. Z's may be preceded by one of the three insertion characters $ + or - and interspersed with the four insertion characters . , 0 or B. Whether these insertion characters affect the result of the editing process depends on the nature of the data.

Suppressing leading zeros and inserting blanks ceases when one of the following conditions exists:

1. When the number of suppressed zeros equals the number of Z's specified in the C field.
2. When the first non-zero digit character in the A field is encountered.
3. When the position in the C field is reached where a decimal point insertion is specified. Zero suppression and blank replacement cannot continue beyond a decimal point, hence, a decimal point is never followed by blanks in an edited field.

    - If either a $ + - is specified before Z's, the character is inserted in the edited data regardless of leading zero suppression.
    - If either a comma, B, or 0 is encountered in the edit field before zero suppression has terminated, the character is not inserted in the edited data, but is suppressed and a blank is inserted.
    - In the special case where the edited data has a value of zero, the entire edited data is replaced by blanks if a 9 or * does not appear in the edit picture. This special rule overrides the condition that zero suppression terminates when a decimal point is encountered. If one of the three insertion characters ($, +, -) is specified but the value of the edited field is zero, a blank is inserted instead of the insertion character.

* The asterisk (*) is specified in the same way and with the same results as the character Z, except that suppressed characters are replaced by asterisks instead of blanks. The rules for the Z character apply also to the use of the asterisk.

$ If one more dollar sign ($) than the number of leading zeros to be suppressed is specified at the left end of a C field, this dollar sign acts as an insertion character. Each of the other dollar signs corresponds to a leading zero to be suppressed. This use of the dollar sign has the same results as described for the Z character, except that the dollar sign is inserted directly preceding the first non-suppressed character. A dollar sign used in this way as a replacement character is known as a floating dollar sign as it virtually "floats" through all of the suppressed characters.

If one or more floating dollar signs are specified in a C field, the edited data always contains a dollar sign whether or not any suppression occurs since one of the dollar signs is an insertion character. Each dollar sign specified in a picture (including the insertion $) is included in determining the character count ($\$_2$) of the edited field.

\+ When a plus sign (+) is used as a replacement character, it functions as a floating plus sign and is specified in the C field one more time than the number of leading zeros to be suppressed. Its function is the same as the floating dollar sign, with the following exception:

> If the A field data is positive or unsigned a plus sign is inserted in the character position directly preceding the first non-suppressed character. If the A field data is negative, a minus sign is inserted in this character position in the edited field.

\- When a minus sign is used as a replacement character it functions as a floating minus sign and is specified in the C field one more time than the number of leading zeros to be suppressed. Its function is the same as the floating dollar sign and floating plus sign with the following exception:

> If the A field data is negative a minus sign is inserted in the character position directly preceding the first non-suppressed character. If the value of the A field data is positive or unsigned, a blank is inserted in this position in the edited data instead of a minus sign.

TABLE 6-9. EDITING EXAMPLES

Field A Data	Field C Editing Data	Resultant Field C Edited Data
4 8	$99	$ 4 8
4 8 3 4	$99.99	$ 4 8 . 3 4
4 8 3 4	9,999	4 , 8 3 4
2 9 2	+999	+ 2 9 2
2 9 $\overline{2}$	+999	- 2 9 2
2 9 $\overline{2}$	999-	2 9 2 -
2 9 2	999-	2 9 2 Δ
2 4 3 2 1	$BB999.99	$ Δ Δ 2 4 3 . 2 1
2 4 3 2 1	$00999.99	$ 0 0 2 4 3 . 2 1
1 1 3 $\overline{4}$	99.99CR	1 1 . 3 4 C R
1 1 3 4	99.99CR	1 1 . 3 4 Δ Δ
2 3 7 $\overline{6}$	99.99DB	2 3 . 7 6 D B
2 3 7 6	99.99DB	2 3 . 7 6 Δ Δ
0 0 9 2 3	ZZ999	Δ Δ 9 2 3
0 0 9 2 3	ZZZ99	Δ Δ 9 2 3
0 0 0 0 0 0	ZZZZ.ZZ	Δ Δ Δ Δ Δ Δ Δ
0 0 9 2 3	$***.99	$ * * 9 . 2 3
0 0 0 8 2 4	$$$$9.99	Δ Δ Δ $ 8 . 2 4
0 0 5 2 $\overline{6}$	---9.99	Δ Δ - 5 . 2 6
3 2 6 5	$$$.99	$ 3 2 . 6 5

UPPERMOST CHARACTER POSITIONS →

THE SIGN IS CONTAINED IN THE LOWEST ORDER CHARACTER. ONLY MINUS SIGNS ARE SHOWN HERE →

THE Δ FIGURE INDICATES A BLANK POSITION →

Field A Data	Field C Editing Data	Resultant Field C Edited Data
0 0 0 0 1 2 3 4	$ZZZ,ZZZ.99	$ Δ Δ Δ Δ Δ 1 2 . 3 4
0 0 1 2 3 4 5 6	$***,**9.99	$ * * 1 , 2 3 4 . 5 6
1 2 3 4 5 6 3 4	$***,***.99	$ 1 2 3 , 4 5 6 . 3 4
0 0 0 0 1 $\overline{2}$	-ZZZ,ZZZ	- Δ Δ Δ Δ Δ 1 2
1 2 3 4 5 6 2 $\overline{4}$	$ZZZ,ZZ9.99CR	$ 1 2 3 , 4 5 6 . 2 4 C R
0 0 1 2 3 4 0 0	$$$$,$$9.99	Δ Δ $ 1 , 2 3 4 . 0 0
0 0 0 0 0 0	$$$$,$$$.99	Δ Δ Δ Δ Δ Δ Δ Δ . 0 0
0 0 0 0 1 2 5 $\overline{6}$	----,---.99DB	Δ Δ Δ Δ Δ - 1 2 . 5 6 D B

1. Only one replacement character of the set Z * $ + and - can be used within a single editing C field even though it may be specified more than once.

2. If one of the replacement characters Z or * is used with one of the insertion characters $ + or -, the plus sign or the minus sign may be specified as either the leftmost or rightmost character in the editing C field.

3. A plus sign and a minus sign may not be included in the same editing C field.

4. A leftmost plus sign and a dollar sign may not be included in the same editing C field.

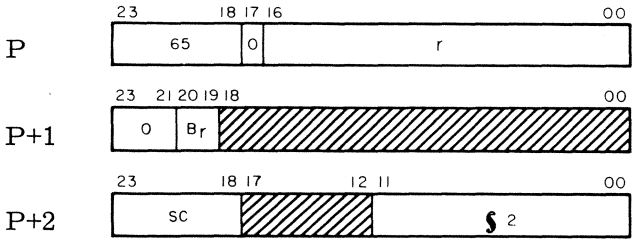
5. A leftmost minus sign and a dollar sign may not be included in the same editing C field.

6. The character 9 may not be specified to the left of a replacement character.

7. Symbols which may appear only once are: decimal point, CR, and DB.

8. The decimal point may not be the rightmost character in an editing C field.

SCAN, LR, EQ
Search Field A
Left to Right for Equality



r = unmodified address of the highest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

sc = 6-bit scan character compared against characters in field A

$§_2$ = number of characters to be searched

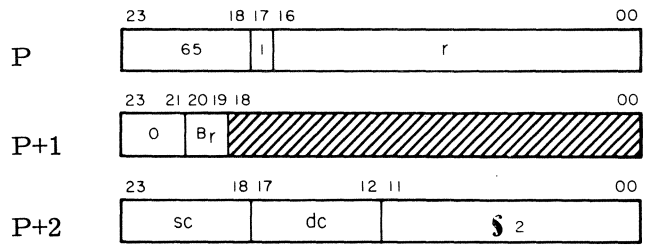
Bits 00 - 18 of P + 1, and bits 12 - 17 of P + 2 should be loaded with zeros.

Instruction Description: Search field A from left to right beginning with the 6-bit character at location R and RNI at P + 4 if a character is found that is equal to the scan character 'sc'. If a character is not found that equals 'sc' after the entire field defined by $§_2$ has been searched, RNI at P + 3.

Comments: If a character comparison occurs during the search, the number of searched characters is transferred to the lower 12 bits of B^3. If an unsuccessful search is made, then (B^3) = $§_2$. The upper 3 bits of B^3 are of no consequence in this instruction. BCD codes of 12_8, 32_8, and 52_8 do not compare equal to zero for this instruction.

The BDP Condition register is not used.

SCAN, LR, EQ, dc
Search Field A Left to Right for Equality, Delimited



r = unmodified address of the highest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

sc = 6-bit scan character compared against characters in field A.

$§_2$ = number of characters to be searched.

dc = 6-bit delimiting character compared against characters in field A

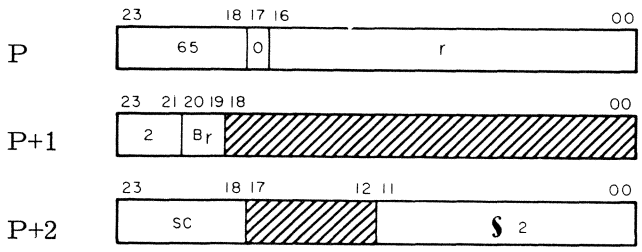
Bits 00 - 18 of P + 1 should be loaded with zeros.

Instruction Description: Search field A from left to right beginning with the 6-bit character at location R and RNI at P + 4 if a character is found that is equal to the scan character 'sc'. If a character is not found that equals 'sc' after the entire field defined by $§_2$ has been searched or if a character is found that equals the delimiting character 'dc', RNI at P + 3. (If 'sc' equals 'dc' and comparison with field A occurs, RNI at P + 4).

Comments: If a character comparison is found during the search, the number of characters searched is transferred to the lower 12 bits of B^3. If an unsuccessful search is made, then $(B^3) = §_2$. The upper 3-bits of B^3 are of no consequence in this instruction. BCD codes of 12_8, 32_8, and 52_8 do not compare equal to zero for this instruction.

The BDP Condition register is not used.

SCAN, LR, NE
Search Field A Left to Right for Inequality



r = unmodified address of the highest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

sc = scan character which is compared against characters in field A.

$\$_2$ = number of characters to be searched

Bits 00 - 18 of P + 1, and bits 12 - 17 of P + 2 should be loaded with zeros.

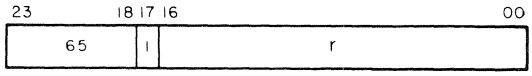
Instruction Description: Search field A from left to right beginning with the 6-bit character at location R, and RNI at P + 4 if a character is found that is not equal to the scan character 'sc'. If a character is not found that is not equal to 'sc' after the entire field defined by $\$_2$ has been searched, RNI at P + 3.

Comments: If an unequal character comparison is found during the search, the number of characters searched is transferred to the lower 12 bits of B^3. If all characters searched equal 'sc', then $(B^3) = \$_2$. The upper 3 bits of B^3 are of no consequence in this instruction. BCD codes of 12_8, 32_8, and 52_8 do not compare equal to zero for this instruction.

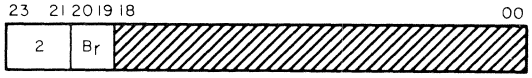
The BDP Condition register is not used.

SCAN, LR, NE, dc Search Field A Left to Right for Inequality, Delimited

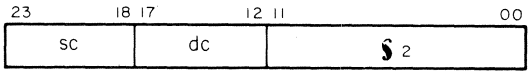
P



P+1



P+2



r = unmodified address of the highest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 3, use index register B^2
If B_r = 0, no indexing

sc = scan character which is compared against characters in field A

$\S_2$ = number of characters to be searched

dc = 6-bit delimiting character compared against the characters in field A

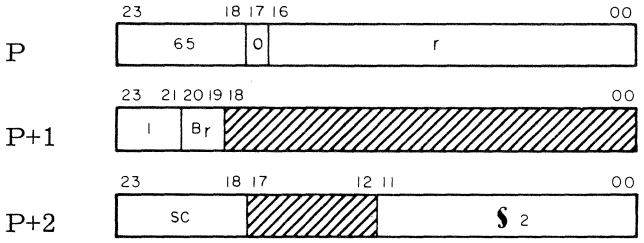
Bits 00 - 18 of P + 1 should be loaded with zeros.

Instruction Description: Search field A from left to right beginning with the 6-bit character at location R, and RNI at P + 4 if a character is found that is not equal to the scan character 'sc'. If no character is found that is unequal to the scan character 'sc' in the entire $\S_2$ field, or if a character is found that is equal to the delimit character 'dc', RNI at P + 3. (If a scan find and a delimit find occur on the same A field character, RNI at P + 4. A delimit find without a scan find can only occur if SC=DC= first A field character. Therefore, a delimit find and a subsequent P + 3 exit can occur only on the first A field character.)

Comments: If an unequal character comparison is found during the search, the number of characters searched is transferred to the lower 12 bits of B^3. If all characters searched equal 'sc', then $(B^3) = \S_2$. The upper 3 bits of B^3 are of no consequence in this instruction. BCD codes of 12_8, 32_8, 52_8 do not compare equal to zero for this instruction.

The BDP Condition register is not used.

SCAN, RL, EQ
Search Field A
Right to Left for Equality



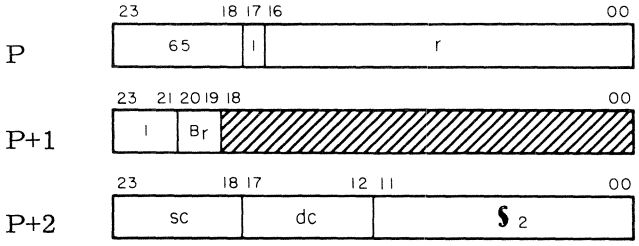
r = unmodified address of the lowest order character in field A.
$R = r + [B_r]$
B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing
sc = scan character which is compared against characters to be searched
$§_2$ = number of characters to be searched
Bits 00 - 18 of P + 1, and bits 12 - 17 of P + 2 should be loaded with zeros.

Instruction Description: Search field A from right to left beginning with the 6-bit character at location R and RNI at P + 4 if a character is found that is identical to the scan character 'sc'. If a character is not found that equals 'sc' after the entire field defined by $§_2$ has been searched, RNI at P + 3.

Comments: If a character comparison is found during the search, the number of characters searched is transferred to the lower 12 bits of B^3. If an unsuccessful search is made, then $(B^3) = §_2$. The upper 3 bits of B^3 are of no consequence in this instruction. BCD codes of 12_8, 32_8, and 52_8 do not compare equal to zero.

The BDP Condition register is set to the sign of field A.

SCAN, RL, EQ, dc
Search Field A Right to Left for Equality, Delimited



r = unmodified address of the lowest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

sc = scan character which is compared against characters in field A

§2 = number of characters to be searched

dc = 6-bit delimiting character compared against the characters in field A

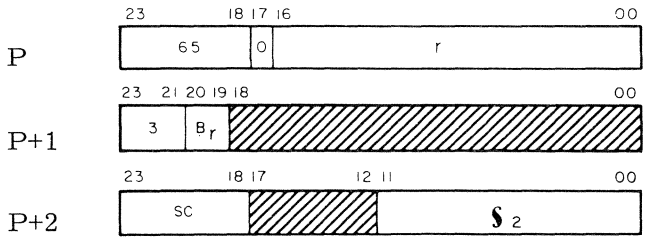
Bits 00 - 18 of P + 1 should be loaded with zeros.

Instruction Description: Search field A from right to left beginning with the 6-bit character at location R and RNI at P + 4 if a character is found that is equal to the scan character 'sc'. If no character is found equal to the scan character 'sc' in the entire §2 field, or if a character is found that equals the delimiting character, RNI at P + 3. (If a scan find and a delimit find occur on the same A field character, RNI at P + 4. A delimit find without a scan find can occur only if SC=DC= first A field character. Therefore, a delimit find and a subsequent P + 3 exit can occur only on the first A field character.)

Comments: If a character comparison is found during the search, the number of characters searched is transferred to the lower 12 bits of B^3. If an unsuccessful search is made, then (B^3) = §2. The upper 3 bits of B^3 are of no consequence in this instruction. BCD codes of 12_8, 32_8, and 52_8 do not compare equal to zero.

The BDP Condition register is set to the sign of field A.

SCAN, RL, NE
Search Field A Right to Left for Inequality



r = unmodified address of the lowest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

sc = scan character which is compared against characters in field A

§2 = number of characters to be searched

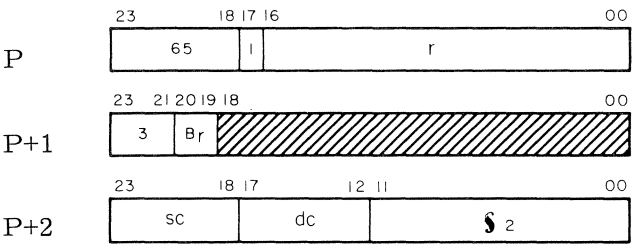
Bits 00 - 18 of P + 1, and bits 12 - 17 of P + 2 should be loaded with zeros.

Instruction Description: Search field A from right to left beginning with the 6-bit character at location R and RNI at P + 4 if a character is found that is not equal to the scan character 'sc'. If a character is not found that is not equal to 'sc' after the entire field defined by §2 has been searched, RNI at P + 3.

Comments: If an unequal character comparison is found during the search, the number of characters searched is transferred to the lower 12 bits of B^3. If all characters searched equal 'sc', then (B^3) = §2. The upper 3 bits of B^3 are of no consequence in this instruction. BCD codes 12_8, 32_8, and 52_8 cause an unequal comparison to zero.

The BDP Condition register is set to the sign of field A.

SCAN, RL, NE, dc Search Field A Right to Left for Inequality, Delimited



r = unmodified address of the lowest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

sc = scan character which is compared against characters in field A

$§_2$ = number of characters to be searched

dc = 6-bit delimiting character compared against the characters in field A

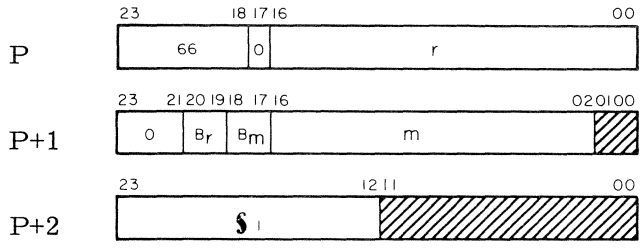
Bits 00 - 18 of P + 1 should be loaded with zeros.

Instruction Description: Search field A from right to left beginning with the 6-bit character at location R and RNI at P + 4 if a character is found that is not equal to the scan character 'sc'. If a character is not found that is not equal to 'sc' after the entire field defined by $§_2$ has been searched, or if a character is found that equals the delimiting character 'dc', RNI at P + 3. (If 'sc' equals 'dc' and an unequal comparison with field A occurs, RNI at P + 4.)

Comments: If an unequal character comparison is found during the search, the number of characters searched is transferred to the lower 12 bits of B^3. If all characters searched equal 'sc', then (B^3) = $§_2$. The upper 3 bits of B^3 are of no consequence in this instruction. BCD codes of 12_8, 32_8, and 52_8 cause an unequal comparison to zero.

The BDP Condition register is set to the sign of field A.

CVDB
Convert BCD to Binary



r = unmodified address of the highest order BCD character in field A.
R = r + [B_r]

B_r = index register flag for field A.
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

m = unmodified address of the highest order bits in binary field C.
M = m + [B_m]

B_m = index register flag for binary field C (same bit functions as B_r). Bits 02-14 of the index register are used for word address indexing (13 bit index-sign extended). Bits 00 and 01 must be set to "1's".

S_1 = number of BCD characters to be converted.

Bits 00 and 01 of P + 1, and bits 00 - 11 of P + 2 should be loaded with zeros.

Instruction Description: Convert a BCD number of up to 14_{10} numeric characters in magnitude (including sign) into its binary equivalent in one's complement notation and store the 48-bit result in locations M and M + 1.

Comments: The sign of the binary number is always stored in bit 23 of M. The maximum positive BCD number (fourteen 9's) is equal to 2657142036437777_8. The complement of this number in M and M + 1 equals a negative field of fourteen 9's.

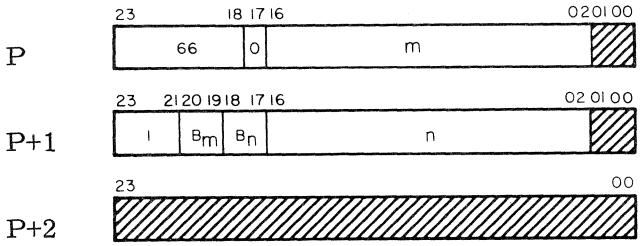
The BDP Condition register is set to the sign of field A. The B^3 register contains a count of the BCD characters converted.

A BCD fault is generated if one of the following conditions occur:

1. Zone portion of any character (except sign character) does not equal zero.
2. Numeric portion of any character in field A contains a BCD code greater than 11_8, except the sign character where a 12_8 code is legal.
3. The sign character contains a 72_8 code.
4. More than 14_{10} BCD characters are specified by S_1.

Operation continues despite any BCD fault.

CVBD
Convert Binary to BCD



m = unmodified address of the highest order bits in binary field A.
M = m + [B_m]

B_m = index register flag for field A.
If B_m = 1 or 3, use index register B^1
If B_m = 2, use index register B^2
If B_m = 0, no indexing
Bits 02-14 of the index register are used for word address indexing (13-bit index-sign extended). Bits 00 and 01 must be set to 1's.

n = unmodified address of the highest order BCD character in field C.

B_n = index register flag for field C (same bit functions as B_m).

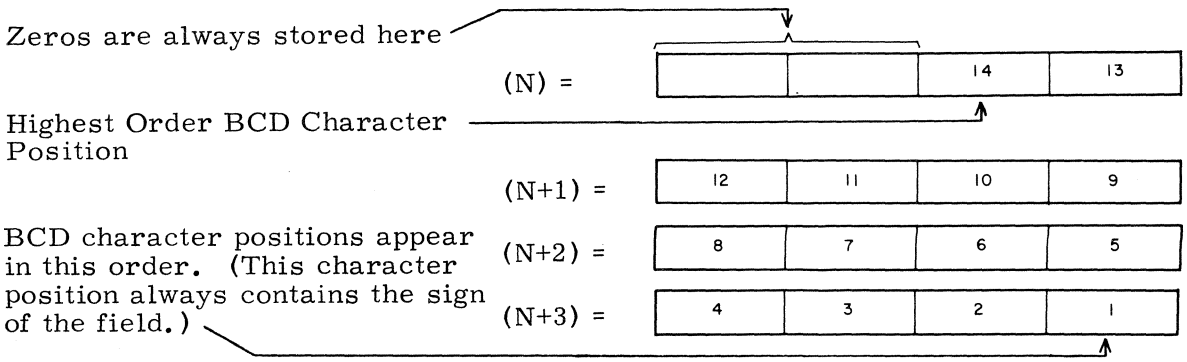
Bits 00 and 01 of P, bits 00 and 01 of P + 1, and bits 00 - 23 of P + 2 should be loaded with zeros.

Instruction Description: Convert the 48-bit binary number (including sign) at M and M + 1 into its signed, numeric BCD equivalent and store in field C.

Comments: (M) and (M + 1) are always analyzed for the binary number to be converted. This is true even if the binary number does not utilize all of the bit positions of M. The sign of the original binary number is located in bit 23 of M. The sign of the binary number is stored in the field C sign character and is set in the BDP Condition register.

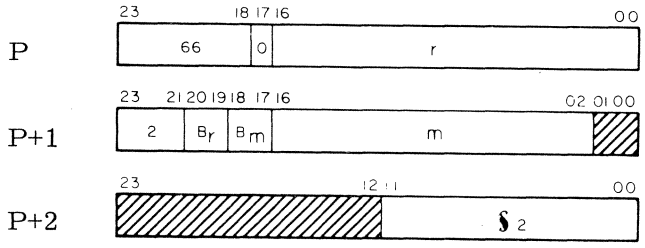
The original contents of B^3 is destroyed and the final contents of B^3 (bits 0-11) is not meaningful.

A BCD fault is generated if the conversion results in a number of more than 14_{10} BCD characters. However, the lower 14_{10} characters will be correct.



DTA
Convert BCD to ASCII

Applicable to 3504 BDP only.



r = unmodified address of the highest order character in BCD field A. R = r + [B_r]

B_r = index register flag for field A. If B_r = 1 or 3, use index register B^1 If B_r = 2, use index register B^2 If B_r = 0, no indexing

m = unmodified word address of the highest order characters in ASCII field C. M = m + [B_m]

B_m = index register flag for field C (same bit functions as B_r). Bits 02-14 of the index register are used for word address indexing (13 bit index-sign extended). Bits 00 and 01 must be set to "1's".

$§_2$ = number of characters to be converted

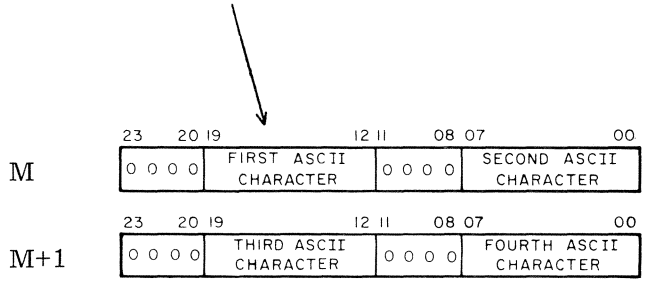
Bits 00 and 01 of P + 1, and bits 12 - 23 of P + 2 should be loaded with zeros.

Instruction Description: Convert from left to right an alphanumeric field of up to 4095 6-bit BCD characters in field A into the 8-bit ASCII characters in field C.

Comments: Refer to Appendix A for a comparative listing of BCD and ASCII codes. BCD codes of 12_8, 32_8, and 52_8 are not treated as zero.

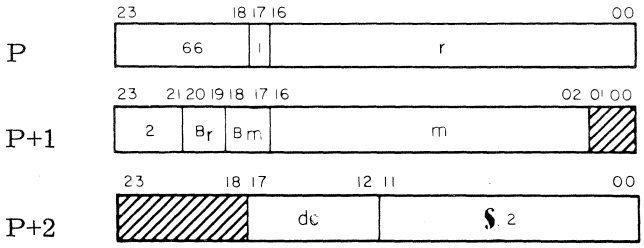
If the last ASCII character is stored in bits 12-19, the remainder of the word is filled with zeros. The count in the B^3 register upon completion equals four times the number of store cycles. The BDP Condition register is set to the sign of field A.

First ASCII character is always placed in this character position of the word specified by M.



DTA, dc
Convert BCD to ASCII, Delimited

Applicable to 3504 BDP only.



r = unmodified address of the highest order character in BCD field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

m = unmodified address of the highest order word in ASCII field C.
M = m + [B_m]

B_m = index register flag for field C (same bit functions as B_r). Bits 02-14 of the index register are used for word address indexing (13-bit index-sign extended). Bits 00 and 01 must be set to ones.

$\$_2$ = number of characters in ASCII field C

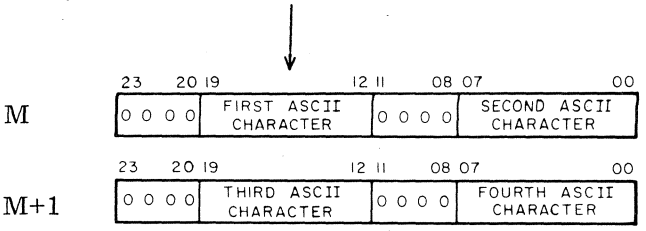
dc = 6-bit delimiting character compared against characters in field A.

Bits 18 - 23 of P + 2, and 00 and 01 of P + 1 should be loaded with zeros.

Instruction Description: Convert from left to right an alphanumeric field of up to 4095 6-bit BCD characters in field A into the 8-bit ASCII characters in field C. The operation is terminated when the number of characters specified by $\$_2$ has been converted or the delimiting character is recognized in field A. A character equal to the delimiting character is converted when encountered.

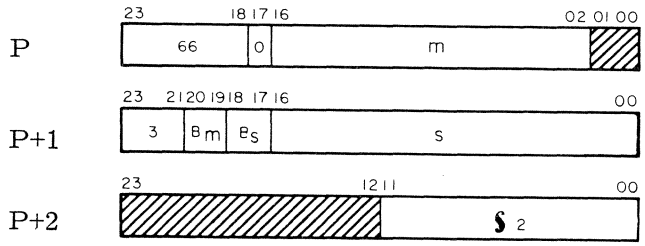
Comments: The BCD codes of 12_8, 32_8, and 52_8 are not treated as zero for this instruction. The count in the B^3 register upon completion equals four times the number of store cycles.

The first ASCII character is always placed in this character position of the word specified by M.



ATD
Convert ASCII to BCD

Applicable to 3504 BDP only.



m = unmodified word address of the highest order characters in ASCII field A. M = m + [B_m] (Refer to diagram below.)

B_m = index register flag for field A
If B_m = 1 or 3, use index register B^1
If B_m = 2, use index register B^2
If B_m = 0, no indexing
Bits 02-14 of the index register are used for word address indexing (13-bit index-sign extended). Bits 00 and 01 must be set to ones.

s = unmodified address of the highest order character in BCD field C. S = s + [Bs]

B_S = index register flag for field C (same bit functions as B_m)

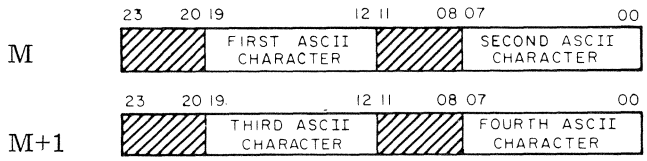
$\S_2$ = number of characters in BCD field C

Bits 00 and 01 of P, and bits 12-23 of P + 2 should be loaded with zeros.

Instruction Description: Convert from left to right a field of up to 4095 8-bit ASCII characters in field A into 6-bit BCD characters in field C. The operation is terminated when $\S_2$ is exhausted.

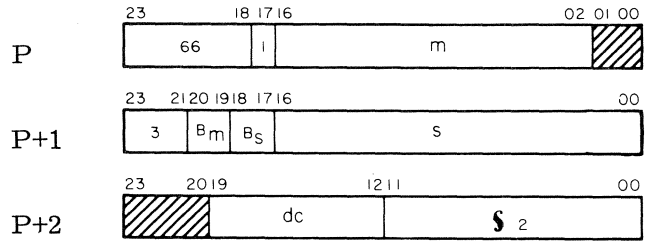
Comments: A BCD fault is generated if bit positions 05 and 06 for any character contain both "1's" or both "0's" (an illegal character). An illegal character is set to zero in field C and conversion continues to completion. The B^3 register indicates the number of BCD characters stored.

The first ASCII character is always located in this character position of the word specified by M. Shaded areas are not used.



ATD, dc
Convert ASCII to BCD, Delimited

Applicable to 3504 BDP only.



m = unmodified word address of the highest order characters in ASCII field A. (Refer to diagram below.)
M = m + [B_m]

B_m = index register flag for field A
If B_m = 1 or 3, use index register B^1
If B_m = 2, use index register B^2
If B_m = 0, no indexing
Bits 02-14 of the index register are used for word address indexing (13-bit index-sign extended). Bits 00 and 01 must be set to "1's".

s = unmodified address of the highest order character in BCD field C.
S = s + [Bs]

B_s = index register flag for field C. (same bit functions as B_m)

$§_2$ = number of characters in BCD field C

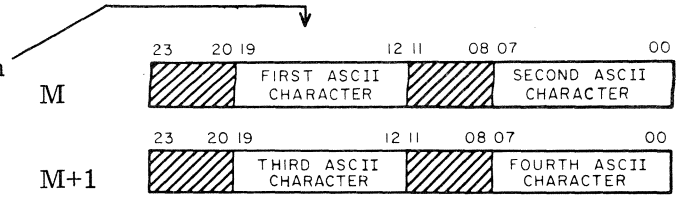
dc = 8-bit delimiting character compared against characters in field A

Bits 20 - 23 of P + 2, and 00 and 01 of P should be loaded with zeros.

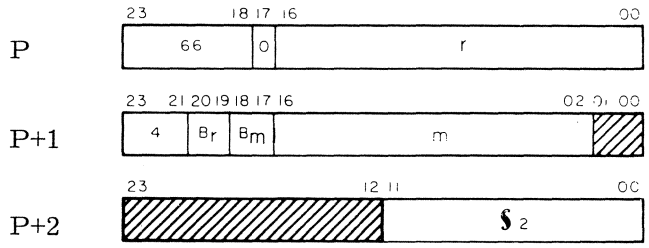
Instruction Description: Convert from left to right a field of up to 4095 8-bit ASCII characters in field A into 6-bit BCD characters in field A. The operation is terminated when $§_2$ is exhausted or an ASCII character equal to the delimiting character is recognized. A character equal to 'dc' is converted when encountered.

Comments: A BCD fault is generated if bit positions 05 and 06 for any character contain both "1's" or both "0's" (an illegal character). An illegal character is set to zero in field C and conversion continues to completion. The B^3 register indicates the number of BCD characters stored.

The first ASCII character is always derived from this character position of the word specified by M. Shaded areas are not used.



PAK
Convert 6-Bit BCD to 4-Bit BCD



r = unmodified address of the lowest order character in the A field.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

m = unmodified address of the lowest order word in field C.
M = m + [B_m]

B_m = index register flag for field C (same bit functions as B_r). Bits 02-14 of the index register are used for word address indexing (13-bit index-sign extended). Bits 00 and 01 must be set to "1's".

$2 = number of 6-bit characters in field A to pack.

Bits 00 and 01 of P + 1, and bits 12 - 23 of P + 2 should be loaded with zeros.

Instruction Description: Convert from right to left, a field of numeric 6-bit BCD characters in field A into 4-bit BCD characters and store the result in field C. The 4-bit BCD characters are stored six per word, with the zone bits of the original 6-bit characters removed.

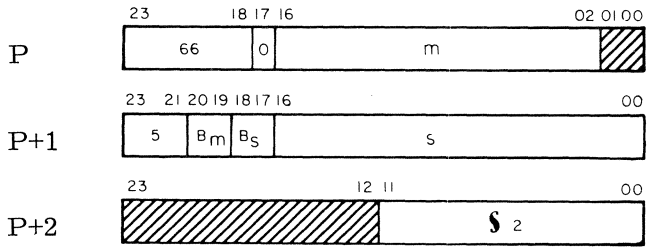
Comments: The sign of field A is converted into a 4-bit sign character (1010_2 - positive, 1011_2 - negative) and placed in the low order character of field C prior to packing BCD characters. Any A field character with a 12_8 - 17_8 code is forced to a zero in field C. A full word store is used. Any unfilled 4-bit characters in the highest order word are stored as zeros. The B^3 register contains a count equal to four times the number of 24-bit words stored. The BDP Condition register is not used.

A BCD fault is generated if one of the following conditions occur:

1. Zone portion of any character (except sign character) does not equal zero.
2. Numeric portion of any character in field A contains a BCD code greater than 11_8, except the sign character where a 12_8 code is legal.
3. The sign character in field A contains a 72_8 code.

Operation continues despite any BCD fault.

UPAK
Convert 4-Bit BCD to 6-Bit BCD



m = unmodified address of the lowest order word in field A.
M = m + [B_r]

B_m = index register flag for field A
If B_m = 1 or 3, use index register B^1
If B_m = 2, use index register B^2
If B_m = 0, no indexing
Bits 02-14 of the index register are used for word address indexing (13-bit index-sign extended).
Bits 00 and 01 must be set to "1's".

s = unmodified address of the lowest order character in field C.
S = s + [Bs]

B_s = index register flag for field C (same bit functions as B_m)

$\$_2$ = number of characters resulting in field C.

Bits 00 and 01 of P, and bits 12 - 23 should be loaded with zeros.

Instruction Description: Covert from right to left a field of packed numeric 4-bit BCD characters in field A into 6-bit BCD characters and store the result in field C. Zone bits of new 6-bit characters are set to "00" except in the lowest order character which receives the algebraic sign from the 4-bit sign character in field A. Field C contains one less character than field A due to the elimination of the 4-bit sign character.

Comments: The conversion commences with the sign character of the 4-bit BCD field. If the sign is positive (1010_2), "00" zone bits are stored in the field C sign character; if negative (1011_2), a "10" is stored. A result sign character code (zone and numeric) of 40_8 is converted to 52_8. Any A field character (other than the sign character) containing a 12_8 - 17_8 code results in a zero in field C.

The B^3 register indicates the number of 6-bit characters stored in field C. The BDP Condition register is not used.

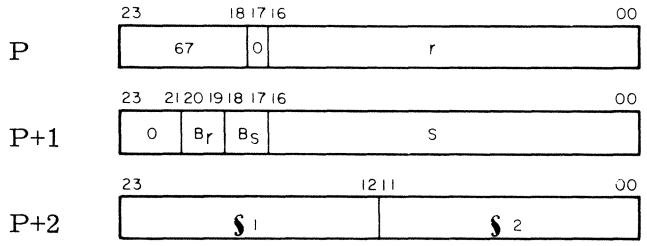
A BCD fault is generated if one of the following conditions occur:

1. The sign character in field A is other than 1010_2 or 1011_2 (the sign of field C is then set to zero), or
2. Any other character in field A contains a 12_8 - 17_8 code, except the second lowest character where a 12_8 code is legal.

Operation continues despite any BCD fault.

ADM
Add Field A to Field C

Applicable to 3504 BDP only. See next page for 3514 BDP.



r = unmodified address of lowest order character in field A. $R = r + [B_r]$

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of lowest order character in field C. $S = s + [B_s]$

B_s = index register flag for field C (same bit functions as B_r)

§1 = 12-bit character count specifying the length of the A field

§2 = 12-bit character count specifying the length of the C field

Instruction Description: Add the BCD contents of field A (addend) to field C (augend) proceeding from right to left. The sum appears in field C. The lowest order character of each field specifies sign. The sign of the sum is contained in the character defined by the original address S.

Comments: Field A may be shorter than field C as carries can be set into progressively higher order positions of field C. This instruction normally terminates when field C is exhausted. However, an early exit occurs when the most significant field A character is processed if all of the following conditions are present:

1. ADD and signs alike
2. No carry
3. The current cumulative arithmetic result is not zero.

If any character position of either field contains a 12_8 - 17_8 code, it is converted to zero before the add operation.

The BDP Condition register indicates a positive sign (00) or negative sign (10) according to the result in field C. A zero result may have either a positive or negative sign.

Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

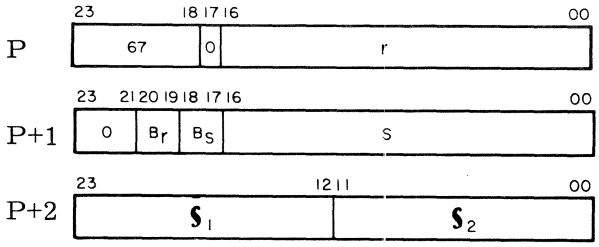
A BCD fault is generated if one of the following conditions occur:

1. An arithmetic carry is attempted out of the upper limit of field C.
2. §1 > §2
3. Zone portion of characters in either field (except sign character) does not equal zero.
4. Numeric portion of characters in either field contains a BCD code greater than 11_8, except the sign character where a 12_8 code is legal.
5. The sign character in either field contains a 72_8 code.

Operation continues despite any BCD fault.

ADM
Add Field A to Field C

Applicable to 3514 BDP only. See previous page for 3504 BDP.



r = unmodified address of lowest order character in field A. $R = r + [B_r]$

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of lowest order character in field C. $S = s + [B_s]$

B_s = index register flag for field C (same bit functions as B_r)

$1 = 12-bit character count specifying the length of the A field

$2 = 12-bit character count specifying the length of the C field

Instruction Description: Add the BCD contents of field A (addend) to field C (augend) proceeding from right to left. The sum appears in field C. The lowest order character of each field specifies sign. The sign of the sum is specified by the character defined by the original address 'S'.

Comments: Field A may be shorter than field C as carries can be set into progressively higher order positions of field C. This instruction normally terminates when the C field is exhausted; however, an early exit occurs prior to this time when the following conditions are present:

3514 - 1. A field exhausted,
2. ADD and signs alike,
3. No carry, and
4. The current cumulative arithmetic result is not zero.

A 60_8 code (blank) encountered in either field is treated as a 00_8 code. A sign character code of 40_8 in the result field is converted to a 52_8, the code for magnetic tape character negative zero. If any character position of either field contains a 12_8 - 17_8 code, it is converted to zero before the add operation.

The BDP Condition register indicates a positive sign (00) or negative sign (10) according to the result in field C. A zero result will always be positive.

Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

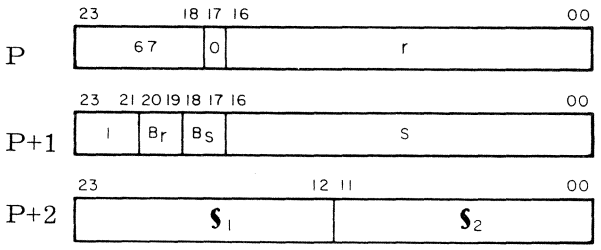
A BCD fault is generated if one of the following conditions occurs:

1. An arithmetic carry is attempted out of the upper limit of field C.
2. $1 > $2
3. Zone portion of characters in either field (except sign character) does not equal zero.
4. Numeric portion of characters in either field contains a BCD code greater than 11_8, except the sign character where a 12_8 code is legal.
5. The sign character in either field contains a 72_8 code.

Operation continues despite any BCD fault.

SBM
Subtract Field A from Field C

Applicable to 3504 BDP only. See next page for 3514 BDP.



r = unmodified address of the lowest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the lowest order character in field C.
S = s + [B_s]

B_s = index register flag for field C (same bit functions as B_r)

$1 = 12-bit character count specifying the length of the A field

$2 = 12-bit character count specifying the length of the C field

Instruction Description: Subtract the BCD contents of field A (subtrahend) from field C (minuend) proceeding from right to left. The difference appears in field C. The lowest order character in fields A and C contain the algebraic sign of their respective fields. The sign of the difference is contained in the character specified by the original address "s".

Comments: $1 must be $\leq$ $2 since field C must accommodate the result of the subtraction. This instruction normally terminates when field C is exhausted. However, an early exit occurs when the most significant field A character is processed if all of the following conditions are present:

1. SUBTRACT and signs alike,
2. No carry, and
3. The current cumulative result is not zero.

If any character position of either field contains a 12_8 - 17_8 code, it is converted to zero before the subtract operation.

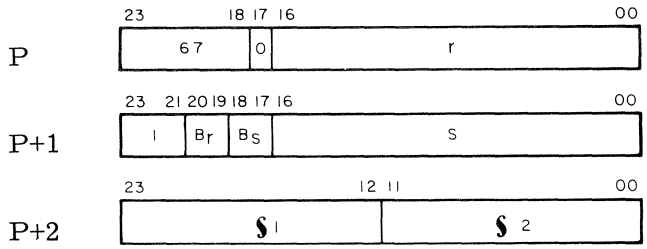
The BDP Condition register indicates a positive sign (00) or a negative sign (10) according to the result in field C. A negative zero result will never occur. BCD codes 12_8 - 17_8 in the lowest order character in either field are converted to zero.

Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

The conditions for generating a BCD fault are identical to those for the ADM instruction.

SBM
Subtract Field A from Field C

Applicable to 3514 BDP only. See previous page for 3504 BDP.



r = unmodified address of the lowest order character in field A.
$R = r + [B_r]$

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the lowest order character in field C.
$S = s + [B_s]$

B_s = index register flag for field C (same bit functions as B_r)

$§_1$ = 12-bit character count specifying the length of the A field

$§_2$ = 12-bit character count specifying the length of the C field

Instruction Description: Subtract the BCD contents of field A (subtrahend) from field C (minuend) proceeding from right to left. The difference appears in field C. The lowest order character of each field specifies sign. The sign of the difference is contained in the character specified by the original address S.

Comments: $§_1$ must be $\leq §_2$ since field C must accommodate the result of the subtraction. This instruction normally terminates when the C field is exhausted; however, an early exit occurs prior to this time when the following conditions exist:

1. A field exhausted,
2. SUBTRACT and signs alike,
3. No carry, and
4. The current cumulative result is not zero.

A sign character code of 40_8 in the result field is converted to a 52_8, the code for magnetic tape character negative zero. If any character position of either field contains a 12_8 - 17_8 code, it is converted to zero before the subtract operation.

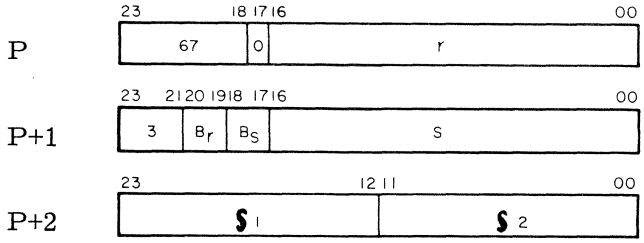
The BDP Condition register indicates a positive sign (00) or a negative sign (10) according to the result in field C. A zero result will always be positive.

Index register B^3 (bits 0-11) records the field C character count as the instruction progresses.

The conditions for generating a BCD fault are identical to those for the ADM instruction.

CMP
Compare Field A to Field C

Applicable to 3504 BDP only. See next page for 3514 BDP.



r = unmodified address of the highest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the highest order character in field C.
S = s + [B_S]

B_S = index register flag for field C (same bit function as B_r)

S_1 = 12-bit character count specifying the length of the A field

S_2 = 12-bit character count specifying the length of the C field

Instruction Description: Compare characters in field A with field C proceeding from left to right. Terminate the operation when an unequal character comparison occurs and RNI at P + 3. If the comparison condition is not satisfied when one of the fields has been completely examined, the remainder of the other field is examined for blanks (60 codes). If the remaining characters are all blanks, the compare operation is terminated as an equal comparison and the next instruction is read from P + 4. If the remainder of the larger field does not contain all blanks and:

If $S_1 > S_2$, then an A > C comparison condition exists.
If $S_2 > S_1$, then an A < C comparison condition exists.

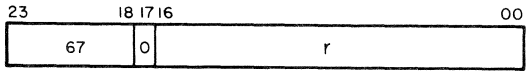
Comments: The result of the comparison is entered into the BDP Condition register as described in the table below. If the fields are unequal, the next instruction is read from P + 3. If the fields are equal, the next instruction is read from P + 4. The count of the C field characters processed is placed in the B^3 register upon completion of the instruction.

Comparison Condition	Contents of BCR
A = C	00_2
A > C	01_2
A < C	10_2

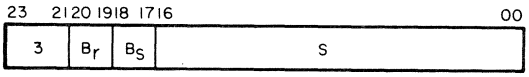
CMP
Collating Compare

Applicable to 3514 BDP only. See previous page for 3504 BDP.

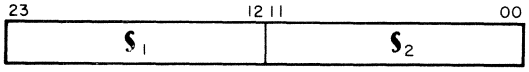
P



P+1



P+2



r = unmodified address of the highest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the highest order character in field C.
S = s + [B_s]

B_s = index register flag for field C (same bit function as B_r)

$1 = 12-bit character count specifying the length of the A field

$2 = 12-bit character count specifying the length of the C field

Instruction Description: Compare characters in field A with field C for inequality, proceeding from left to right. Characters are compared in internal machine code (6-bit BCD). If character inequality is detected, a table in storage (see comments) is referenced to determine high-low status of the unequal characters. Results of the compare are recorded in the Condition register, as follows:

Comparison Condition	Contents of BCR
A = C	00_2
A > C	01_2
A < C	10_2

If field lengths are different and one field is depleted without finding inequality, the remainder of the longer field is examined for blanks (60_8 codes). If all blanks are found, an equal comparison (A=C) is recorded. If a non-blank is detected and

$\S_2 > \S_1$, then an A < C condition is recorded

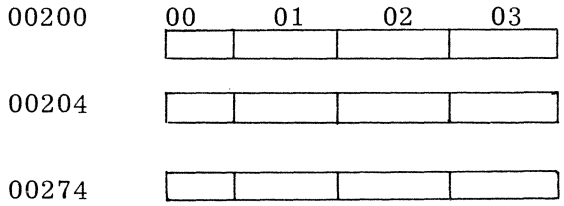
$\S_1 > \S_2$, then an A > C condition is recorded

For inequality (unequal comparison condition), a RNI is performed at P+3; otherwise the next instruction is fetched from P+4.

A count of the C field characters processed is placed in B3 upon completion.

Comments: This instruction permits the internal machine comparison of data fields required in collating (ordering) to be extended to character sets with different collating sequences, such as USASCII. The internal 6-bit BCD code of the 3500 is used for comparing characters; a pre-loaded table in central storage is referenced whenever character inequality between fields is detected to obtain the corresponding relative collating position codes for the two characters in the subject character set. A compare is then made of the two position codes to determine their high-low status.

The table is composed of absolute character locations 00200 - 00277 (words 00040 - 00057) in Page 0 of central storage, a space large enough for 77 relative collating positions.



CAUTION

The table in Page 0 must be available for reading during the execution of this instruction, or an illegal write will occur.

Example

Assume two fields of equal length, to be ordered according to the USASCII collating sequence pre-stored in the storage table:

Field A = 3N374	03	45	03	07	04	BCD code
Field C = 3N3*2	03	45	03	54	02	BCD code

Step 1 - A character by character compare is performed; inequality between the two fields is detected at the 4th character. In BCD, this would indicate an C > A condition; however, examine the table.

Step 2 - The table is referenced to obtain the two corresponding USASCII collating positions for the inequal characters (7 and *). The table addressing is automatically done by indexing the table base address by the 6-bit BCD code:

Field A table character address = 00200 + 07 = 00207

Field C table character address = 00200 + 54 = 00254

00040 (200 - 203)	20	21	22	23
00041 (204 - 207)	24	25	26	27
00053 (254 - 257)	12	03	74	36

NOTE

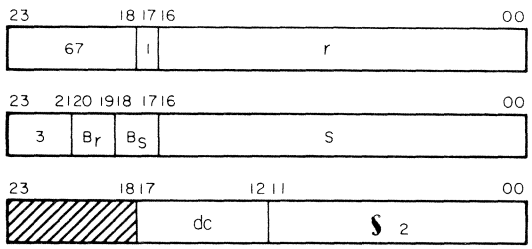
The table contains pre-stored USASCII collating position characters. If the two characters read from this table are equal, a BCD fault is set and a 00_2 code is placed in the Condition register.

Comparing the two USASCII position characters results in a A > C comparison condition, opposite from the BCD ordering. This condition would then be recorded in the condition register as (01_2).

Step 3 - Perform an RNI at P + 3. B3 register would contain a character count of 4 upon the exit.

CMP, dc
Compare Field A to Field C, Delimited

Applicable to 3504 BDP only. See next page for 3514 BDP.



r = unmodified address of the highest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the highest order character in field C.
S = s + [B_S]

B_S = index register flag for field C (same bit functions as B_r)

dc = 6-bit delimiting character

$2 = 12-bit character count specifying the length of field C (and of field A)

Field A always contains the same number of characters as field C during a delimited compare operation. Bits 18 through 23 of P + 2 should be loaded with zeros.

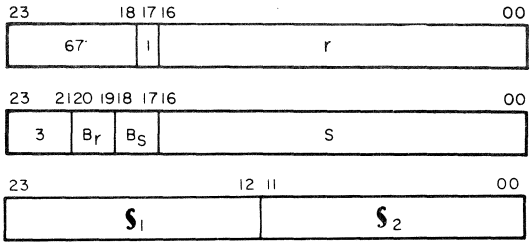
Instruction Description: Compare characters in field A with field C proceeding from left to right. Terminate the operation when: 1) an unequal character comparison occurs (the magnitudes of the unequal characters are noted - see table below), 2) all of the characters in the fields have been examined, 3) a character in either the A or C field equals the delimiting character. RNI at P + 3 for an unequal field comparison; RNI at P + 4 for an equal field comparison.

Comments: The count of the characters processed is placed in the B^3 register upon completion of the instruction. The following table describes the state of the BDP Condition register when the operation is terminated:

Terminating Comparison Condition	Contents of BCR
A = C (through entire field or through delimiting character)	00_2
A > C	01_2
A < C	10_2

CMP, n
Compare Field A to Field C, Numeric

Applicable to 3514 BDP only. See previous page for 3504 BDP.



r = unmodified address of the lowest order character in field A.
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

s = unmodified address of the lowest order character in field C.
S = s + [B_S]

B_S = index register flag for field C (same bit functions as B_r)

$1 = 12-bit character count specifying the length of field A

$2 = 12-bit character count specifying the length of field C

Instruction Description: Compare numeric characters in field A with field C, proceeding from right to left. The operation terminates upon completion of the longer field. If field lengths are different, the remainder of the longer field is compared to zero.

For inequality (unequal comparison condition), a RNI is performed at P+3, otherwise the next instruction is fetched from P+4.

After completion of the compare, the high, low, equal status is contained in the Condition register:

Condition	Code
A = C	00_2
A > C	01_2
A < C	10_2

If the signs of the two fields are unlike, an exit is made with B^3 (lower 12 bits) equal to zero, otherwise B^3 contains the character count of the C field characters processed.

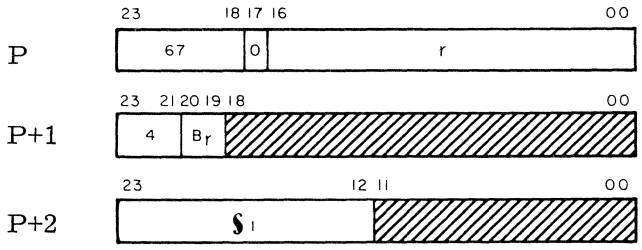
Comments: Positive numbers are considered greater than negative, except for ±0 where they are considered equal. Any illegal number positions of characters (lower 4 bits) are converted to zero and compared as such.

The BCD Fault is generated if one of the following conditions occur:

1. Zone portion of any character (except sign character) does not equal zero.
2. Numeric portion of any character contains BCD code greater than 11_8, except the sign character where a 12_8 code is legal.
3. The sign character contains a 72_8 code.

Operation continues despite any BCD fault.

TST
Test Field A for Sign



r = unmodified address of the lowest order BCD character in field A
R = r + [B_r]
B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing
$\$_1$ = number of BCD characters in field A to be tested
Bits 00 - 18 of P + 1, and bits 00 - 11 of P + 2 should be loaded with zeros.

Instruction Description: Examine the sign character of field A and record the results in the BDP Condition register (see Comments). Field A is scanned from right to left until the first non-zero character is encountered, or until $\$_1$ is exhausted.

Comments: If field A contains all zeros, and the sign character is recognized as zero (contains BCD code of 60, 52, 40, 32, 20, 12, or 00), the Condition register is set to "00". If field A does not equal zero, the Condition register is set as follows:

Field A	Contents of BDP Condition Register
Positive - upper 2 bits of sign character are 00_2, 01_2, or 11_2	01_2
Negative - upper 2 bits of sign character are 10_2	10_2

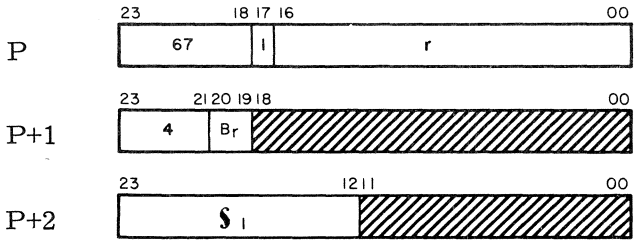
A BCD fault is generated if one of the following conditions occur:

1. Zone portion of any character (except sign character) does not equal zero.
2. Numeric portion of any character contains a BCD code greater than 11_8, except the sign character where a 12_8 code is legal.
3. The sign character contains a 72_8 code.

Operation continues despite any BCD fault.

Index register B^3 (bits 0-11) is set to all zeros.

TSTN
Test Field A for Numeric



r = unmodified address of the lowest order BCD character in field A
R = r + [B_r]

B_r = index register flag for field A
If B_r = 1 or 3, use index register B^1
If B_r = 2, use index register B^2
If B_r = 0, no indexing

$\$_1$ = number of BCD characters in field A to be tested.

Bits 00 - 18 of P + 1, and bits 00 - 11 of P + 2 should be loaded with zeros.

Instruction Description: Examine the sign character of field A and record the results in the BDP Condition register (see Comments). Field A is scanned from right to left until $\$_1$ is exhausted. If the BCD Fault is not set upon completion of the instruction, field A is numeric. The BCD Fault can be sensed by instruction.

Comments: If field A contains all zeros and the sign character is recognized as zero (contains BCD code of 60, 52, 40, 32, 20, 12, or 00), the Condition register is set to "00". If field A does not equal zero, the Condition register is set as follows:

Field A	Contents of BDP Condition Register
Positive - upper 2 bits of sign character are 00_2, 01_2, or 11_2	01_2
Negative - upper 2 bits of sign character are 10_2	10_2

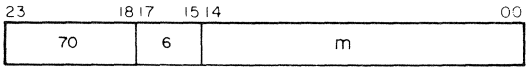
A BCD fault is generated if one of the following conditions occur:

1. Zone portion of any character (except sign character) does not equal zero.
2. Numeric portion of any character contains a BCD code greater than 11_8, except the sign character where a 12_8 code is legal.
3. The sign character contains a 72_8 code.

Operation continues despite any BCD fault.

Index register B^3 (bits 0-11) is set to all zeros.

LBR
Load BDP



Applicable to 3504 BDP only. See next page for 3514 BDP.

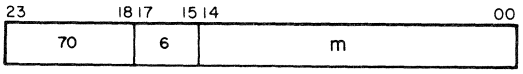
m = storage address. Indexing not permitted.

Instruction Description: Load the BCR (Condition register) and set interrupt recovery conditions within the BDP as defined by (m). The 24 bits of (m) have the following significance:

POSITION	FUNCTION
00 and 01	Contents of the BCR
02	Edit flag indicating a (DB) or (CR) character detected.
03	Zero suppression operation in progress.
04	Edit flag indicating a floating sign ($, +, -) operation is in progress.
05	Edit flag indicating a $ sign is forced.
06	Edit flag indicating a + sign is forced.
07	Edit flag indicating an * sign is forced.
08	Edit flag indicating a floating character is forced.
09	Operand equals zero
10	Signs of operands unlike on ADM or SBM, or incorrect on EDIT.
11	Interrupt occurred during BDP operation.
12 - 23	Number of characters or words for Field A already processed.

Comments: The LBR instruction should be used only when returning from an interrupt routine. The instruction is trapped if the BDP MODE switch is OFF, except during Monitor state when it is a No-Op.

LBR
Load BDP



Applicable to 3514 BDP only. See previous page for 3504 BDP.

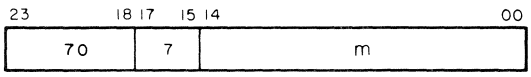
m = storage address. Indexing not permitted.

Instruction Description: Load the BCR (Condition register) and set interrupt recovery conditions within the BDP as defined by (m). The 24 bits of (m) have the following significance:

POSITION	FUNCTION
00 and 01	Contents of the BCR
02	Edit flag indicating a (DB) or (CR) character detected
03	Zero suppression operation in progress.
04	Edit flag indicating a floating sign ($, +, -) operation is in progress.
05	Edit flag indicating a $ sign is forced.
06	Edit flag indicating a + sign is forced or compare flag indicating field sign.
07	Edit flag indicating an * sign is forced or compare flag indicating A > C condition.
08	Edit flag indicating a floating character is forced or compare flag indicating C > A condition.
09	Operand equals zero.
10	Signs of operands unlike on ADM or SBM, or incorrect on EDIT.
11	Interrupt occurred during BDP operation.
12 - 23	Number of characters or words for field A already processed.

Comments: The LBR instruction should be used only when returning from an interrupt routine. The instruction is trapped if the BDP MODE switch is OFF, except during Monitor state when it is a No-Op.

SBR
Store (BDP)



m = storage address. Indexing not permitted.

Instruction Description: Store various operating conditions from within the BDP section at address 'm'. Refer to the LBR instruction for the bit functions of (m).

Comments: Execution of this instruction does not clear the operating conditions within the BDP. This instruction is trapped if the BDP MODE switch is OFF, except during Monitor state when it is a No-Op.

7. CONSOLE AND COMPUTER MAINTENANCE PANEL

GENERAL INFORMATION

The basic 3500 Computer system includes a desk console. The console for the 3504-1 is slightly different than the console for 3514 central processors. The differences are described later in this section. Figure 7-1 shows the desk console for the 3514 central processors. Either console enables the computer operator to control computer operations, perform I/O operations via the on-line typewriter, and to monitor computer and I/O status through visual displays. This section describes register and status displays, operator's switches and controls, and the console typewriter. Also included is a description of the computer maintenance panel.

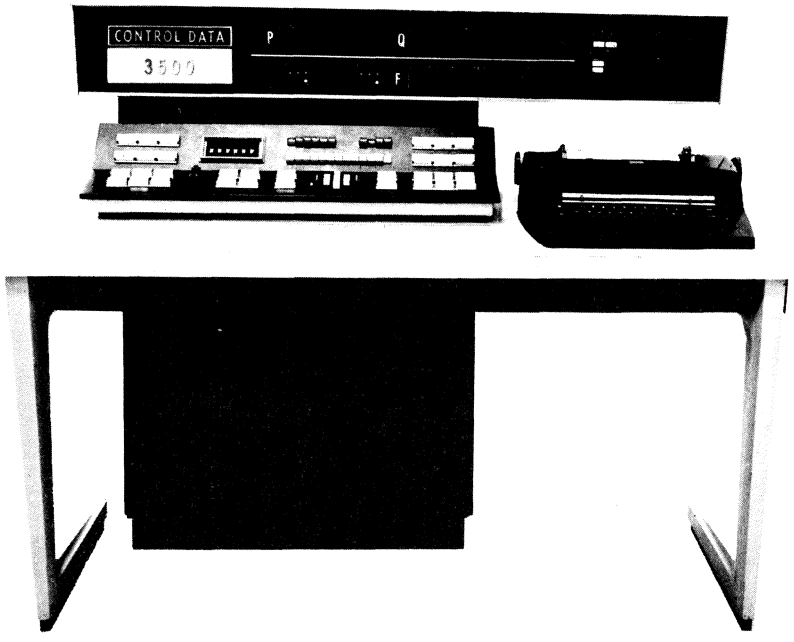


Figure 7-1. 3500 Desk Console

CONSOLE

Register Displays

The operational registers described in Section 1 are selectively displayed in the upper console display area shown in Figure 7-2. Registers are selected for display by switches located on the console switch panel. Three of the displays are shared by registers as follows:

1. 15-bit display -- shared by the Index registers, P register, Last Jump Address register, and Channel Index register.
2. 24-bit display -- shared by the A, Q, E upper, and E lower registers.
3. 24-bit display -- shared by the F and C registers (not selected by switch).

The leftmost segment of each shared display indicates the register that is currently selected and being displayed.

The F register is displayed whenever the keyboard is off and the computer is stopped. The C register is displayed during register entry, sweep and enter, and Read/Write Sto operations.

Transient conditions may not be seen in the displays due to the response time of the indicators.

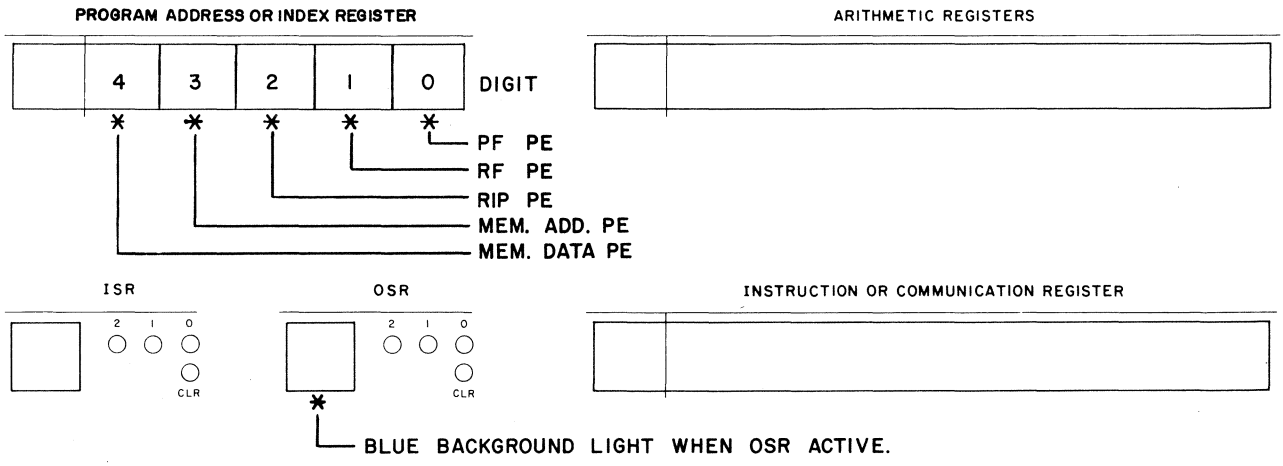


Figure 7-2. Register Displays

Register Entry

Data is entered into the A, Q, P, and B_b registers by first selecting the register for entry on the switch panel and then selecting data with the keyboard number switches. Data is temporarily held in the C register during register entry. An active blue digit-indicator light is superimposed upon each digit in the C register as the number switches are pressed. If an error is made during data entry, pressing the KYBD CLR switch clears only the C register, leaving other registers unaltered. When correct entry into C has been accomplished, pressing the TRANSFER switch gates the data to the selected register. Pressing the TRANSFER switch, with zeros in the C register, enables zeros to the selected register. Data is entered into the ISR and OSR with the binary entry

switches provided near the respective display. These entry switches may be pressed simultaneously or individually.

Data Interchange Display

The data interchange display, shown in Figure 7-3, enables the console operator to monitor the status of each of the eight I/O channels that may be present in a system. Each channel has its own set of READ, WRITE, REJECT, INTERRUPT, and CHANNEL PARITY ERROR indicators (see Table 7-1 for descriptions). Transient conditions may not be seen on the display due to the response time of the indicators.

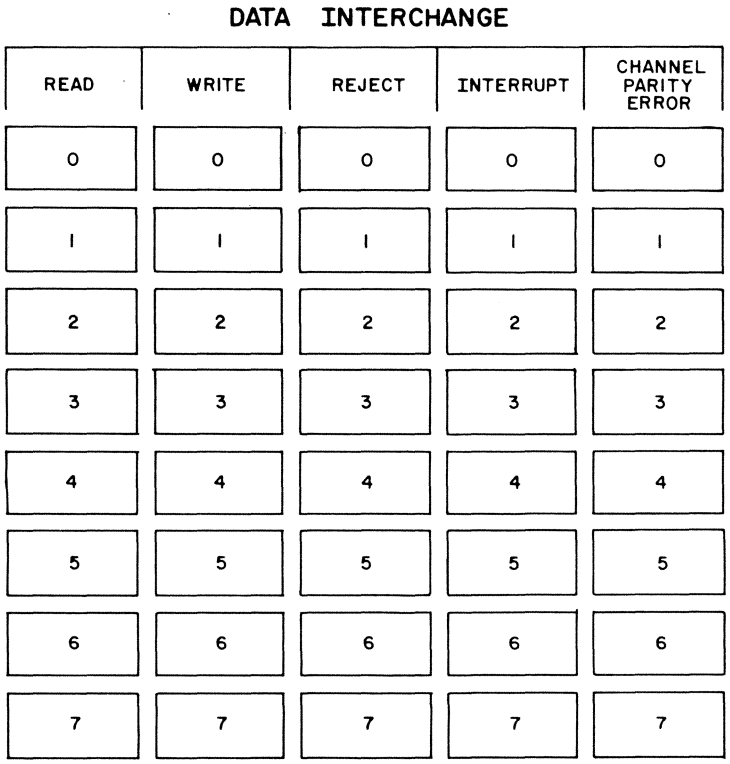


Figure 7-3. Data Interchange Display

TABLE 7-1. DATA INTERCHANGE INDICATOR DESCRIPTIONS

INDICATOR NAME	FUNCTION
READ	Glows when data is being received by the computer on the channel indicated.
WRITE	Glows when data is being transmitted by the computer on the channel indicated.
REJECT	Glows when a Reject signal is received from a peripheral equipment on the channel indicated.
INTERRUPT	Glows when an interrupt is received from a peripheral equipment on the channel indicated. Indicator glows until the interrupting condition is cleared.
CHANNEL PARITY ERROR	Glows when a transmission parity error has occurred on the channel indicated. Indicator glows until the condition is recognized.

Computer Status Display

The computer status display provides the operator with visual indications of the internal conditions of the computer. Status available to the operator generally includes computer operating status and internal fault conditions. Figure 7-4 shows the arrangement of the indicators on the display; the function of each indicator is described in Table 7-2.

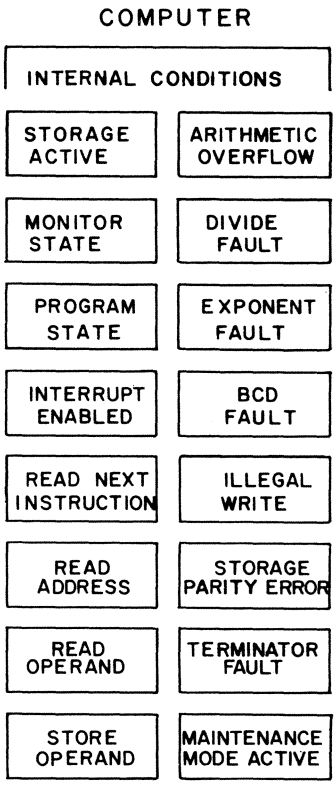


Figure 7-4. Computer Status Display

TABLE 7-2. COMPUTER STATUS INDICATOR DESCRIPTIONS

INDICATOR NAME	FUNCTION
STORAGE ACTIVE	Indicates a storage reference is in progress. Indicator is common to all storage modules.
MONITOR STATE	Indicates the computer is operating in the Monitor state of Executive mode.
PROGRAM STATE	Indicates the computer is operating in the Program state of Executive mode.
INTERRUPT ENABLED	Indicates the interrupt system has been enabled by instruction.
READ NEXT INSTRUCTION	Indicates the computer is reading the next instruction of the program it is currently executing. Usually referred to as the RNI cycle.

TABLE 7-2. COMPUTER STATUS INDICATOR DESCRIPTIONS (Cont'd)

INDICATOR NAME	FUNCTION
READ ADDRESS	Indicates the computer is reading the lower 18 bits of a storage location to form a new address during indirect addressing. Usually referred to as the RADR cycle.
READ OPERAND	Indicates the computer is reading a 24-bit operand from storage for use with the instruction being executed. Usually referred to as the ROP cycle.
STORE OPERAND	Indicates the computer is storing a 24-bit operand somewhere in storage. Usually referred to as the STO cycle.
ARITHMETIC OVERFLOW	Indicates the capacity of the adder has been exceeded. Its capacity, including sign, is 24 or 48 bits for 24-bit precision or 48-bit precision, respectively.
DIVIDE FAULT	The divide fault indicates a quotient, including sign, exceeds 24 or 48 bits for 24-bit precision or 48-bit precision, respectively. Attempts to divide by too small a number, including positive and negative zero, result in a divide fault. During floating point division, a divide fault occurs if division by zero or by a number that is not in floating point format is attempted. If the divisor is not properly normalized a divide fault may also occur. Refer to Appendix B for a description of normalization.
EXPONENT FAULT	Indicates either an exponent overflow ($> +1777_8$) or an exponent underflow ($< -1777_8$) has occurred during a floating point arithmetic operation.
BCD FAULT	Indicates a BCD Fault has occurred within the optional BDP section or a SBCD (77.72) instruction has been executed. Refer to Section 4, BCD Fault, for additional information.
ILLEGAL WRITE	Normally indicates an attempt has been made to write into a protected storage location. Refer to Section 4, Illegal Write Status, for specific information.
STORAGE PARITY ERROR	Indicates a storage address parity error, a storage data parity error, a no-response condition, or a RIP parity error occurred. This indicator is not affected by external I/O parity errors. A Storage Address Parity Error condition also lights the background light for the low order digit in the P register. This light enables the operator to distinguish between address and data parity errors.
TERMINATOR FAULT	Indicates that the internal terminator power supplies are not functioning properly.
MAINTENANCE MODE ACTIVE	Indicates the key-operated maintenance switch on the back control panel of the console is turned ON.

Console Switch Panel

The switches and controls used by the operator to communicate with the computer are located on the console switch panel shown in Figure 7-5. Included on the panel are switches for selecting system operating conditions, setting interrupt and jump conditions, performing maintenance functions, and controlling program execution. The switches used during on-line typewriter operations, located in the upper-right corner of the panel, are discussed later in this section.

NOTE

The RIP MODE and FP MODE switches are on consoles for 3514 processors only.

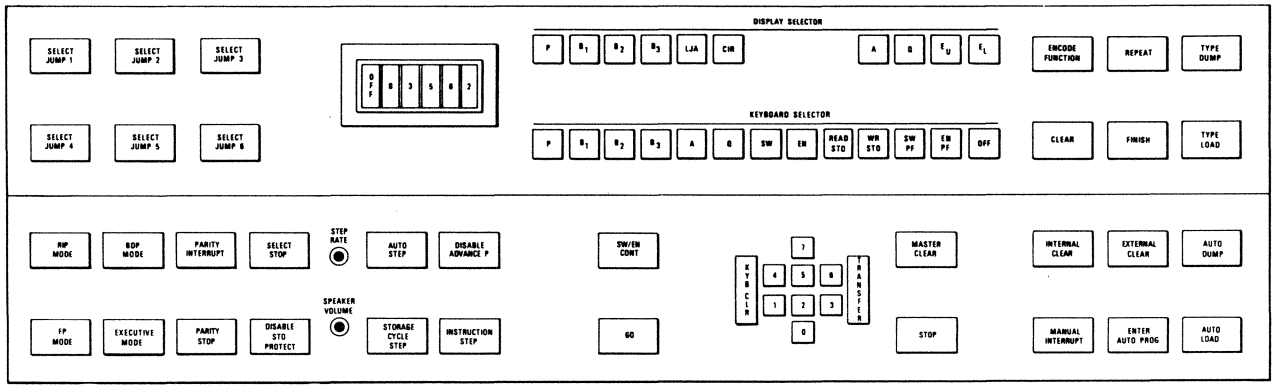


Figure 7-5. Console Switch Panel

Keyboard and Register Display Switches

The group of switches used chiefly for manual register operations and direct execution control are located in the center of the switch panel. The top row of switches (P, B_1, B_2, ,..., E_L) are only used to select registers for display, as described in Table 7-3.

The keyboard switches (used to manually enter and retrieve data from the computer) are described in Table 7-4. Examples of keyboard switch functions are given later in this section. The keyboard is disabled when the computer has stopped because of a BPO compare (see Breakpoint Switch Description in this section) or after a ROP, STO, or RADR cycle when doing Storage Cycle Step operations.

TABLE 7-3. REGISTER DISPLAY SWITCHES

SWITCH NAME	FUNCTION
P	Causes the 15-bit P register to be displayed during GO mode. P is always displayed when the computer is stopped.
B_1, B_2, or B_3	Causes the selected 15-bit Index register to be displayed if the computer is stopped.
LJA	Causes the basic 15-bit storage address of the last jump instruction to be displayed if the computer is stopped.
CIR	Causes the 3-bit Channel Index register to be displayed in the lowest octal digit of the 15-bit display area, provided the computer is stopped.
A	Causes the 24-bit A register to be displayed if the computer is stopped.
Q	Causes the 24-bit Q register to be displayed if the computer is stopped.
E_U, E_L	Causes either the upper or lower half of the 48-bit E register to be displayed if the computer is stopped.

TABLE 7-4. KEYBOARD SWITCH DESCRIPTIONS

SWITCH NAME	FUNCTION
P	Selects the P register for manual keyboard entry.
B_1, B_2, or B_3	Selects one of the index registers for manual keyboard entry.
A	Selects the A register for manual keyboard entry.
Q	Selects the Q register for manual keyboard entry.
SW	Permits instructions to be read from storage and displayed in the C register, without being executed. The sweep begins with the address specified by P; pressing the TRANSFER switch, advances the address in P (except on first transfer). If AUTO STEP is active, the transfer rate is selected with the STEP RATE control.
EN	Permits data to be manually entered into storage while the computer is stopped. Data placed in the C register is transferred to the storage location specified by P as the TRANSFER switch is pressed. P advances to the next address as TRANSFER is pressed. If AUTO STEP is active, the transfer rate is selected with the STEP RATE control.
READ STO	Permits the contents of the storage or register file location specified by the Breakpoint switch to be displayed in the C register at a rate determined by the STEP RATE control. This is possible during Go mode or when the computer is stopped.
WR STO	Permits keyboard entry into the storage or register file location specified by the Breakpoint switch. This can be done during Go mode or when the computer is stopped; the Write occurs as the TRANSFER switch is pressed.
SW PF	Permits page indexes from consecutive locations of the optional Page Index File to be displayed in the C register. The address of the index is specified in the lower 7 bits of the P register. The index is displayed and P advances to the next address (except on the first transfer) as the TRANSFER switch is pressed. If AUTO STEP is active, the transfer speed is selected with the STEP RATE control.
EN PF	Permits data to be manually entered into the page index of the optional Page Index File specified by the lower 7 bits of P. The data enters the index and P advances as TRANSFER is pressed. If AUTO STEP is active, the transfer speed is selected with the STEP RATE control.
OFF	Disables all keyboard controls.
KYBD CLR	Clears the C register.
0 through 7	Numeric switches used for entering data into the C register.
TRANSFER	Transfers data from the C register to a selected register, or into a selected storage, register file, or page index file location.

Condition Switches

The switches used for setting or clearing conditions within the computer, thereby instructing the computer to take some immediate or subsequent action, are described in Table 7-5.

TABLE 7-5. CONDITION SWITCH DESCRIPTIONS

SWITCH NAME	FUNCTION
SELECT JUMP (1-6)	Switches are used to condition selective jump instructions (00. 1-6).
RIP* MODE	Enables the optional Real-Time Interrupt Processor. Refer to Section 4 for information regarding the RIP.
FP* MODE	Enables the double precision multiplication/division and all Floating Point instructions. If the switch is OFF, these instructions are trapped. Refer to Section 4 for a list of trapped instructions.
BDP MODE	Enables the optional Business Data Processing section of the Central Processor to execute a business oriented set of instructions. If the switch is not active, the BDP instruction set is trapped. Refer to Section 4 for a list of trapped instructions.
EXECUTIVE MODE	Permits the computer to operate in the Executive mode, the initially operating state of which is Monitor state. Press the switch during Executive mode to return computer to Non-Executive mode.
PARITY INTERRUPT	Causes the computer to interrupt when a Storage Parity Error or No-Response condition becomes active.
PARITY STOP	Causes the computer to stop when a Storage Parity Error or No-Response condition becomes active.
SELECT STOP	Stops the computer when the SLS (77. 70) instruction is read from memory. When the Go mode is again entered, the computer resumes with the next instruction.
DISABLE STO PROTECT	Disables the protection given by the two sets of Storage Protect switches, but does not affect the protected Auto Load/Auto Dump or program protected areas.
AUTO STEP	Permits INSTRUCTION STEP to be executed at a rate controlled by the STEP RATE control. Also allows speed of Sweep or Enter operations to be controlled by the STEP RATE control.
STORAGE CYCLE STEP	Enables the operator to step through an instruction one storage cycle at a time; that is: RNI, ROP, RADR, or STO.
DISABLE ADVANCE P	Prevents the P register from being incremented. If the GO switch is pressed, the same instruction is repeatedly executed. **
INSTRUC-TION STEP	Enables the operator to execute a program, instruction by instruction. One instruction is executed each time the INSTRUCTION STEP switch is pressed; when the computer stops, the next instruction to be executed is displayed in the F register. Any programmed stop instructions are displayed twice.

* 3514 processor console only

** The P Register will advance to P+1 or P+2 when executing a Pause or Sense instruction.

TABLE 7-5. CONDITION SWITCH DESCRIPTIONS (Cont'd)

SWITCH NAME	FUNCTION
SW/EN MODE	Enables continuous Sweep or Enter operations with storage or the optional Page Index File without using the TRANSFER switch. With the desired operation selected, execution begins when the SW/EN CONT switch is pressed.
GO	Starts program execution at the address specified by the P register. Not used for Sweep or Enter operations.
MASTER CLEAR	Performs both an internal and external clear. The switch is disabled when the computer is in Go mode. If a hang up condition occurs where STOP will not halt the computer, simultaneously press STOP and MASTER CLEAR to restore initial conditions.
STOP	Stops the computer at the end of the current instruction.
INTERNAL CLEAR	Clears internal registers and conditions, including the typewriter controls.
EXTERNAL CLEAR	Clears all external equipment and the I/O channels.
MANUAL INTERRUPT	Forces the computer to automatically enter an interrupt if in Go mode. mode. If the switch is pressed when the computer is stopped, press GO to cause the computer to interrupt. The interrupt system must first be enabled.
ENTER AUTO PROGRAM	Disables storage protection for the Auto Load/Auto Dump areas, enabling programs to be written in these areas.
AUTO DUMP	As this switch is pressed, the computer automatically jumps to address 77760 if in Non-Executive mode, or to address 03740 if in Executive mode, and executes the instruction there. Refer to Section 3 for details.
AUTO LOAD	When this switch is pressed, the computer automatically jumps to address 77740 if in Non-Executive mode, or to address 03700 if in Executive mode, and executes the instruction there. Refer to Section 3 for details.

Breakpoint Switch

The Breakpoint switch is a six-section, eight-position, thumb-wheel switch. The left-hand wheel selects the operating mode, and the other five wheels specify a register file or storage address. There are four mode positions on the Mode Selector switch with an OFF position between each mode; these modes are BPI, BPO, REG, and STO. Breakpoint switch examples appear later in this section.

BPI and BPO Modes: The address from Main Control (before optional address relocation) is continually compared with the instruction or operand address specified by the Breakpoint digit switches. When the Mode Selector switch is set to BPI the computer stops if these values become equal during an RNI sequence. When the Mode Selector switch is set to BPO, the computer stops if these values become equal during an ROP or STO sequence.

REG and STO Modes: In these modes, the operator may either monitor the contents of a register file location or storage address specified by the thumb-wheel digit switches,

or he may store a word in these locations. To monitor a storage location:

1. Set the mode selector to REG (register file location) or STO (storage).
2. Set the Breakpoint switch to the desired register number or storage address.
3. Press the READ STO switch on the keyboard.
4. Adjust the Step Rate control to vary the display rate.

The register or storage contents are repeatedly displayed in the Communication register at the selected repetition rate until another keyboard button is pressed to release READ STO. To write a word in storage:

1. Set the mode selector to REG or STO.
2. Set the Breakpoint switch to the desired register number or storage location.
3. Press the WR STO switch on the keyboard.
4. Enter data into the Communication register by pressing the numeric switches and finally the TRANSFER switch.

If the computer is in Go mode, the data is written into the desired storage or register file location at the end of the instruction currently being executed. Pressing another Register Select or Mode Select switch releases the WR STO operation.

Speaker Volume and Step Rate Controls

The console loudspeaker is mounted under the console table; its volume is controlled by a knob on the switch panel. The loudspeaker receives its input from the upper 3 bits of the A, Q, or E register, whichever is selected. Sound is produced when one of these bits is toggled at an audio frequency.

The STEP RATE control is used for regulating the speed of READ STO and Auto Step operations, and can also control sweep and enter speeds if the AUTO STEP switch is active. The control gives an adjustable range of approximately 1 to 550 cycles per second.

Emergency Off

The red momentary switch located between the display and switch panels is used to remove power from the whole computer system in case of fire or other such extreme emergency. It should not be used for a normal power shutdown.

Examples of Switch Operations

Keyboard Switch Functions

A. To enter data into the A register:
   1. Press the A register entry switch on the keyboard
   2. Enter all eight digits of the Communication register by pressing the appropriate numeric key switches*

*If all eight digit positions of the Communication register are not entered before the TRANSFER switch is pressed, zeros will be entered into the remaining digit positions.

3. Press the TRANSFER switch
4. Press the KEYB CLR switch

B. To enter data into the Q register:

1. Press the Q register switch and repeat step A (2 through 4)

C. To enter the Program Address counter (P register) with a specific address:

1. Press the P register
2. Enter the lower five digits of the Communication register by pressing the appropriate numeric key switches
3. Press the TRANSFER switch
4. Press the KEYB CLR switch

D. To enter an operand at a specific address:*

1. Perform step C.
2. Press the EN switch
3. Enter all eight digits of the Communication register by pressing the appropriate numeric key switches
4. Press the TRANSFER switch
5. The count in the P register has now incremented by one. If data is to be entered into this memory location, repeat step D (3 and 4) for as many succeeding entries as required
6. Press the KEYB CLR switch when all data has been entered into the successive group of memory locations

E. To read an operand from a specific storage address:

1. Perform step C
2. Press the SW switch
3. Press the TRANSFER switch
4. The contents of the specified storage address are now displayed in the Communication register. (The count in the P register is not incremented when the TRANSFER switch is initially pressed.)
5. If the TRANSFER switch is again pressed, the P register is incremented by one, and the contents of the new address are displayed.
6. Press the KEYB CLR switch when all of the desired memory locations within a successive group have been examined.

NOTE

Step E only permits the operator to examine the contents of specific storage locations. The instructions are not executed during this operation.

* The Breakpoint switch may be used in lieu of this operation. (Refer to Example D, Figure 7-6.)

F. To enter zeros or another operand into all storage locations:

1. Press the EN switch
2. Enter all eight digits of the Communication register by pressing the appropriate numeric key switches
3. Press the SW/EN CONT switch. This switch remains engaged until STOP is pressed.
4. Press the STOP switch
5. Press the KEYB CLR switch

G. The following procedure is applicable for sweeping storage during certain maintenance routines:

1. Press the SW switch
2. Press the SW/EN CONT switch. This switch remains engaged until the STOP switch is pressed
3. Press the STOP switch
4. Press the KEYB CLR switch

NOTE

Steps H through K cannot be executed unless the optional address relocation feature of the Relocation section is present.

H. To enter a 12-bit operand into a specific page index:

1. Set P to a specific PIF address (000-177) as outlined in step C. (Only the lower 7 bits of P are recognized.)
2. Press the EN PF switch
3. Enter the lower four digits of the Communication register by pressing the appropriate numeric key switches.
4. Press the TRANSFER switch
5. The PIF address in the P register has now incremented by one. If data is to be entered into this PIF location, repeat step H (3 and 4) for as many succeeding entries as required.
6. Press the KEYB CLR switch when all data has been entered into the successive group of PIF locations.

I. To read an index from the PIF:

1. Perform step H, 1.
2. Press the SW PF switch
3. Press the TRANSFER switch
4. The specified index of the PIF is now displayed in the lower 12 bits of the Communication register. (The P register is not incremented when the TRANSFER switch is initially pressed.)
5. If the TRANSFER switch is again pressed, the P register is incremented by one and the index of the new PIF address is displayed.

6. Press the KEYB CLR switch when all of the desired indexes within a successive group have been examined.

J. To enter zeros or another operand into all indexes of the PIF:

1. Press the EN PF switch
2. Enter the lower four digits of the Communication register by pressing the appropriate numeric key switches.
3. Press the SW/EN CONT switch. This switch remains engaged until the STOP switch is pressed.
4. Press the KEYB CLR switch

K. The following procedure is applicable for sweeping all indexes of the PIF during certain maintenance routines:

1. Press the SW PF switch
2. Press the SW/EN CONT switch. This switch remains engaged until the STOP switch is pressed.
3. Press the STOP switch
4. Press the KEYB CLR switch.

Condition Switch Functions

A. To enter a special routine into the Non-Executive mode Auto Load storage area:

1. Press the MASTER CLEAR switch
2. Holding down the keyboard STOP switch, press the AUTO LOAD switch. Release both switches. The P register should now read 77740. (Holding the STOP switch down prevents the computer from entering the Go mode and executing the previous Auto Load routine.)
3. Press the ENTER AUTO PROG switch
4. Press the keyboard EN switch
5. Enter the first instruction of the new routine at address 77740 by pressing the appropriate numeric key switches.
6. Press the keyboard TRANSFER switch
7. Repeat step A (5 and 6) for addresses 77741 through 77757
8. Press the MASTER CLEAR switch. This clears the registers and cancels the ENTER AUTO PROG function.
9. Press the KEYB CLR switch.

B. To enter a special routine into the Non-Executive mode Auto Dump storage area:

Repeat step A (1 through 9) using the AUTO DUMP switch and filling the storage area covered by addresses 77760 through 77777.

C. To execute the Auto Load routine:

1. Press the MASTER CLEAR switch
2. Press the AUTO LOAD switch. The computer automatically executes the Auto Load routine and stops when a stop or halt instruction is recognized. The Auto Load function is automatically cleared when the first I/O operation is completed.

D. To execute the Auto Dump routine:

Perform step C but use the AUTO DUMP switch instead of the AUTO LOAD switch.

E. To execute a program at Auto Step rate:

1. Set the P register to the first address of the program to be executed
2. Press the AUTO STEP switch
3. Adjust the STEP RATE display control
4. Press the AUTO STEP switch again to cancel the function and stop program execution. The only way to exit from the Auto Step mode is to press the AUTO STEP switch again. In the Auto Step mode, halt and jump instructions are executed, but the computer does not stop. Program execution is not affected by pressing the STOP switch. The computer continues cycling through memory until the AUTO STEP switch is again pressed.

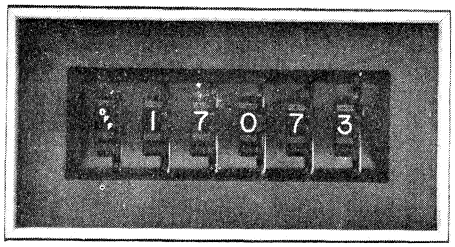
NOTE

To load or execute a subroutine in the Auto Load or Auto Dump areas while in Executive mode, perform the same operations as for Non-Executive mode except that the addresses for the respective areas will be as follows:

Auto Load: 003700 through 003737

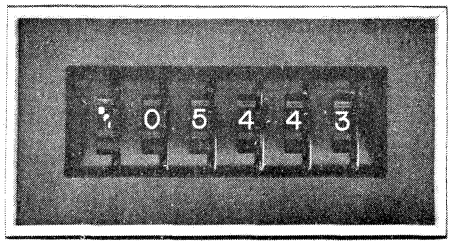
Auto Dump: 003740 through 003777

EXAMPLE A



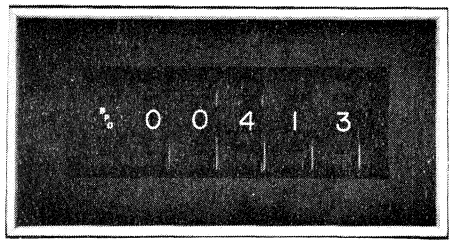
The Breakpoint switch is inoperative whenever an OFF designator is displayed. An OFF designator separates the REG, STO, BPI and BPO positions.

EXAMPLE B



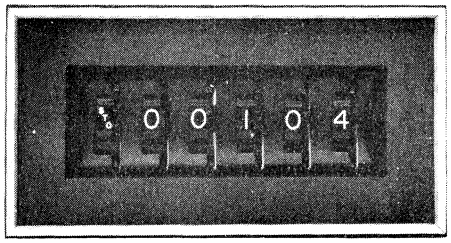
During the normal execution of a program, the computer stops when a RNI is attempted at memory location 05443. A jump to this location also causes the computer to stop. If the program references memory location 05443 for an operand, the computer ignores the Breakpoint switch.

EXAMPLE C



The computer stops only when an attempt is made to read or store an operand at address 00413.

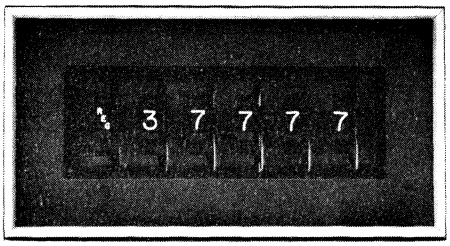
EXAMPLE D



If the WR STO switch on the keyboard is pressed and data has been entered into the Communication register, the data is transferred to memory location 00104 when the TRANSFER switch is pressed.

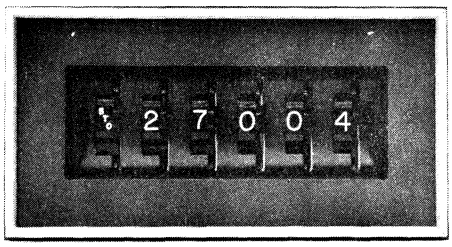
Figure 7-6. Breakpoint Switch Examples

EXAMPLE E



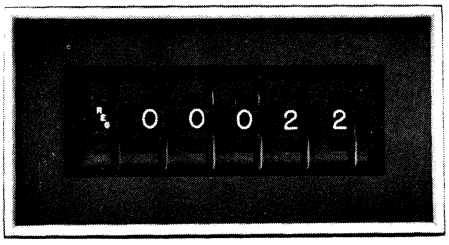
If the WR STO switch on the keyboard is pressed and data has been entered into the Communication register, the data will be transferred to register 77 when the TRANSFER switch is pressed. (Only the lower two digits are recognized when the designator switch is in the REG position. The programmer must use caution when writing into the Register File to prevent destruction of other data. Refer to Section 1, Table 1-1.)

EXAMPLE F



If the READ STO switch on the keyboard is pressed, the contents of memory location 27004 are displayed in the Communication register at a repetition rate determined by the Step Rate control. (If the memory location depicted by the Breakpoint switch exceeds the storage capacity of the system, the computer selects the address that corresponds to the storage capacity of the system.)

EXAMPLE G



If the READ STO switch on the keyboard is pressed, the contents of register 22 are displayed in the Communication register at a repetition rate determined by the Step Rate control. (Only the lower two digits are of consequence when the REG designator is displayed. In this case register 22, the Real-Time Clock, is being referenced.)

Figure 7-6. Breakpoint Switch Examples (Cont'd)

Typewriter

The console typewriter is used as an I/O device to enter character blocks into storage and to type-out selected character blocks from storage. Data is converted to internal BCD code by typewriter logic and is then transmitted. Typewriter I/O is controlled by Block Control in much the same way as normal channel I/O with storage; data is held in the Block Control D4 register for synchronization during transfers between the typewriter and storage. Typewriter requests are honored by Block Control according to the priority listed in Section 1.

Two storage addresses define the limits of the data block and therefore must be stored in the Register File prior to an input or output operation. Register 23 contains the program state number and the initial character address of the block. Register 33 contains the last character address, plus one (refer to the CTO and CTI instructions for registers 23 and 33 operand formats). Because the initial character address is incremented for each storage reference, it always shows the address of the character currently being stored or dumped. Output operations occur at the rate of 15 characters per second. Input operations are limited by the operator's typing speed.

The console typewriter control switches are shown in Figure 7-7 and their functions are described in Table 7-6.

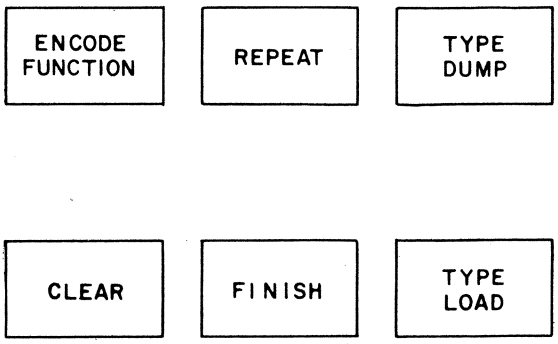


Figure 7-7. Console Typewriter Switches

The following procedure should be followed when doing typewriter input or output:

1. Check status
2. Set registers 23 and 33 of the Register File to the appropriate addresses
3. Set tabs, margins and spacing; turn on typewriter
4. Clear
5. Type out or type in

TABLE 7-6 CONSOLE TYPEWRITER SWITCHES AND INDICATORS

SWITCH	SWITCH (S) INDICATOR (I)	FUNCTION
ENCODE FUNCTION	S/I	This switch enables the typewriter to send to storage the special function codes for backspace, tab, carriage return, upper-case shift, and lower-case shift.
TYPE LOAD	S/I	This switch allows the computer to receive a block of input data from the typewriter. The TYPE LOAD indicator remains on until either FINISH, REPEAT, or CLEAR is pressed, or until the last character of the block has been stored. If the program immediately reactivates the typewriter, it may appear that the light does not go off.
TYPE DUMP	S/I	This switch causes the computer to send data to the typewriter for printout. It is a momentary contact switch that is illuminated until the last character in the block has been printed or CLEAR is pressed.
REPEAT	S/I	This switch is pressed during a Type Load operation to indicate that a typing error occurred. This switch deactivates busy sense line 10 (see PAUS instruction). If the computer does not respond, this light remains on.
FINISH	S/I	This switch is pressed during a Type Load operation to indicate that there is no more data in the current block. This action is necessary if the block that the operator has entered is smaller than the block defined by registers 23 and 33. This switch also deactivates busy sense line 09. If the computer does not respond, this light remains on.
CLEAR	S/I	This switch clears the typewriter controls and sets the typewriter to lower case but does not cancel the ENCODE FUNCTION switch.

Status Checking

The programmer may wish to check the status of the typewriter before proceeding. This is done with the Pause instruction. Status response is returned to the computer through two status lines.

The typewriter control transmits two status signals that are checked by the Pause Sensing Mask using the Pause instruction. These status signals are:

Bit 09 Type Not Finish

Bit 10 Type Not Repeat

An additional status bit appears on sense line 08. This code is Type Busy and is transmitted by Block Control when a typewriter operation has been selected. If the programmer is certain of the status of the typewriter, this operation may be omitted.

Set Registers 23 and 33

Registers 23 and 33 define the limits of the typewriter I/O operation. These registers are set by instruction or by entering the registers using the Breakpoint switch.

Set Tabs, Margins, and Spacing

All tabs, margins, and paper spacing must be set manually prior to the input or output operation. A tab may be set for each space on the typewriter between margins.

Clear

There are three types of Clears which may be used to clear all conditions (except Encode Function) existing in the typewriter control. These are:

1. Internal Clear or a Master Clear

   This signal clears the typewriter control and sets the typewriter to lower case.

2. Clear Channel, Search/Move Control, or Type Control instruction (77.51).

   This instruction selectively clears a channel, the S/M control, or, by placing a "1" in bit 08 of the instruction, the typewriter control, and sets the typewriter to lower case.

3. Clear Switch on Typewriter

   This switch clears the typewriter control and sets the typewriter to lower case.

Type In and Type Load

Executing the CTI (77.75) instruction or pressing the TYPE LOAD switch on the console or typewriter permits the operator to enter data directly into storage from the typewriter. When the TYPE LOAD indicator on the console glows, the operator may begin typing. The ENCODE FUNCTION switch must be pressed to enable backspace, tab, carriage return, and case shifts to be transmitted to the computer during a typewriter input operation.

Input is in Character mode only. As each character is typed, the information is transmitted via the Data Bus to the storage address specified by Block Control. This address is incremented as characters are transmitted. When the current address equals the terminating address, the TYPE LOAD indicator goes off and the operation is terminated. Data is lost if the operator continues typing after the TYPE LOAD indicator goes off.

Type Out and Type Dump

The typewriter begins to type out when the computation section executes a CTO (77.76) instruction, or when the operator presses the TYPE DUMP switch on the console. Single 6-bit characters are sent from storage to the typewriter via the lower 6 bits of the Data Bus. When the current address equals the terminating address, the TYPE DUMP indicator goes off and the operation is terminated.

During a Type Out operation, the keyboard is locked to prevent loss of data in the event a key is accidentally pressed.

Table 7-7 lists the internal BCD codes, typewriter printout and upper-or lower-case shift that applies to the console typewriter. All character transmission between the computation section and the typewriter is in the form of internal BCD. The typewriter logic makes the necessary conversion to the machine code.

NOTE

Shifting to upper case (57) or lower case (32) is not necessary except on keyboard symbols where both upper and lower cases are available. The standard type set has two sets of upper case letters and no lower case letters. This eliminates the need for specifying a case shift.

TABLE 7-7 CONSOLE TYPEWRITER CODES

PRINT-OUT	CASE	INTERNAL BCD CODE	PRINT-OUT	CASE	INTERNAL BCD CODE
0	L	00	-	L	40
1	L	01	J	U or L	41
2	L	02	K	U or L	42
3	L	03	L	U or L	43
4	L	04	M	U or L	44
5	L	05	N	U or L	45
6	L	06	O	U or L	46
7	L	07	P	U or L	47
8	L	10	Q	U or L	50
9	L	11	R	U or L	51
±	U	12	° (degree)	U	52
=	L	13	$	U	53
"	U	14	*	U	54
:	U	15	#	U	55
;	L	16	%	U	56
?	U	17	(Shift to UC)		57
+	U	20	(Space)		60
A	U or L	21	/	L	61
B	U or L	22	S	U or L	62
C	U or L	23	T	U or L	63
D	U or L	24	U	U or L	64
E	U or L	25	V	U or L	65
F	U or L	26	W	U or L	66
G	U or L	27	X	U or L	67
H	U or L	30	Y	U or L	70
I	U or L	31	Z	U or L	71
(Shift to LC)		32	&	U	72
.	U and L	33	,	U and L	73
)	U	34	(	U	74
'	L	35	(Tab)		75
@	U	36	(Backspace)		76
!	L	37	(Carriage Return)		77

*L = Lower Case; U = Upper Case

COMPUTER MAINTENANCE PANEL

Figure 7-8 is an illustration of the computer maintenance panel. Physical location of the panel is indicated in Figure 1-1.

Operational displays on the panel monitor the Page File Address and PP designators from the original address, and display the contents of the Page Index that is referenced during relocation. The upper 6 bits of the final storage address is displayed in parallel with the two sets of Storage Protect switches. The storage protect feature is described in detail in Section 2.

Maintenance Controls

The switches located in the bottom two rows of the panel are for maintenance purposes only, and are operative only when the key-operated maintenance switch on the back control panel of the console is turned ON. With this switch OFF, Maintenance mode and Test mode are disabled, the Real-Time Clock is operating, and all Timing Margins are normal.

The Channel Coupler permits testing one or more I/O channels independent of peripheral equipment. A channel can act as a peripheral equipment to another channel; the coupler hardware provides Reply, Reject, and End-of-Record signals as necessary to control connect, function, and core-to-core transfer operations.

By setting bits in the Maintenance register, one can control the cycling of certain arithmetic instructions and can store the contents of registers and networks normally unavailable to the program. The register is used during Maintenance mode and is loaded with a 54.0 instruction. A complete description of the Maintenance mode and Channel Coupler features can be found in the 3500 Customer Engineering Manual.

The Test Mode switch provides continuous execution of a GO-STOP-MASTER CLEAR program at a rate controlled by the Step Rate control on the console switch panel.

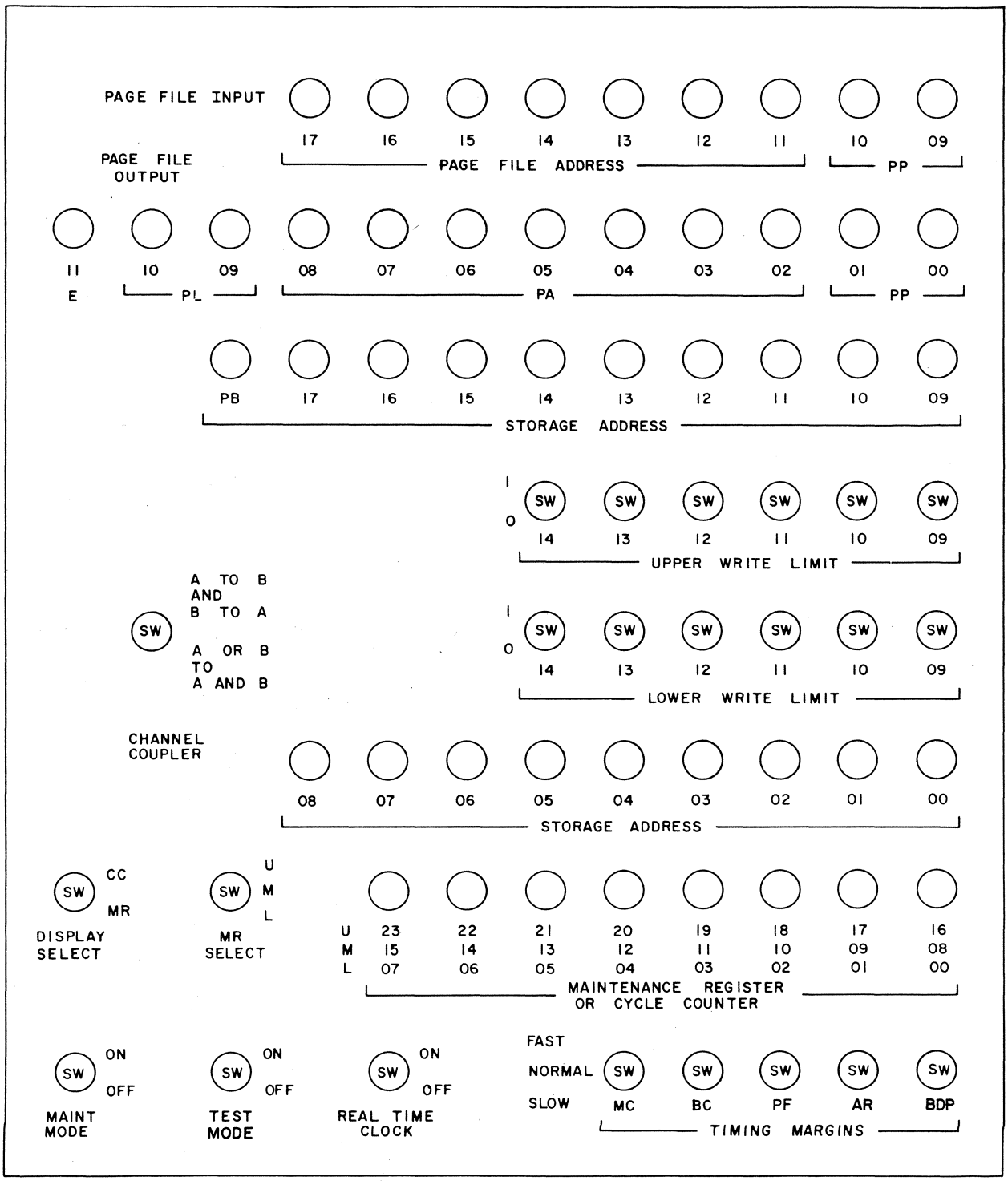


Figure 7-8. Computer Maintenance Panel

APPENDIX A

CONTROL DATA 3100, 3200, 3300, 3500 COMPUTER SYSTEMS CHARACTER SET AND BCD/ASCII CODE CONVERSIONS

CONTROL DATA 3100, 3200, 3300, 3500 COMPUTER SYSTEMS
CHARACTER SET

INTERNAL BCD CODES	EXTERNAL BCD CODES	CONSOLE TYPEWRITER CHARACTERS (USES INTERNAL BCD ONLY)	MAGNETIC TAPE UNIT CHARACTERS	PUNCHED CARD CODES
00	12	0 (zero)	0 (zero)	0
01	01	1	1	1
02	02	2	2	2
03	03	3	3	3
04	04	4	4	4
05	05	5	5	5
06	06	6	6	6
07	07	7	7	7
10	10	8	8	8
11	11	9	9	9
12	(illegal)	±	---	2, 8
13	13	=	#	3, 8
14	14	"	@	4, 8
15	15	:	---	5, 8
16	16	;	---	6, 8
17	17	?	(file mark)	7, 8
20	60	+	&	12
21	61	A	A	12, 1
22	62	B	B	12, 2
23	63	C	C	12, 3
24	64	D	D	12, 4
25	65	E	E	12, 5
26	66	F	F	12, 6
27	67	G	G	12, 7
30	70	H	H	12, 8
31	71	I	I	12, 9
32	72	(Shift to lower case)	+0	12, 0
33	73	.(period)	.	12, 3, 8
34	74	)	⌑	12, 4, 8
35	75	'(apostrophe)	---	12, 5, 8
36	76	@	---	12, 6, 8
37	77	!	---	12, 7, 8

(Continued on next page)

CHARACTER SET (Cont'd)

INTERNAL BCD CODES	EXTERNAL BCD CODES	CONSOLE TYPEWRITER CHARACTERS (USES INTERNAL BCD ONLY)	MAGNETIC TAPE UNIT CHARACTERS	PUNCHED CARD CODES
40	40	-(minus)	-(minus)	11
41	41	J	J	11, 1
42	42	K	K	11, 2
43	43	L	L	11, 3
44	44	M	M	11, 4
45	45	N	N	11, 5
46	46	O	O	11, 6
47	47	P	P	11, 7
50	50	Q	Q	11, 8
51	51	R	R	11, 9
52	52	° (degree)	-0	11, 0
53	53	$	$	11, 3, 8
54	54	*	*	11, 4, 8
55	55	#	---	11, 5, 8
56	56	%	---	11, 6, 8
57	57	(Shift to upper case)	---	11, 7, 8
60	20	(space)	(blank)	(blank)
61	21	/	/	0, 1
62	22	S	S	0, 2
63	23	T	T	0, 3
64	24	U	U	0, 4
65	25	V	V	0, 5
66	26	W	W	0, 6
67	27	X	X	0, 7
70	30	Y	Y	0, 8
71	31	Z	Z	0, 9
72	32	&	---	0, 2, 8
73	33	, (comma)	, (comma)	0, 3, 8
74	34	(	%	0, 4, 8
75	35	(tab)	---	0, 5, 8
76	36	(backspace)	---	0, 6, 8
77	37	(carriage return)	---	0, 7, 8

BCD/ASCII CONVERSION TABLE

6-BIT BCD CODE	8-BIT ASCII CHARACTER	BINARY STATUS OF ASCII CHARACTER (BIT POSITIONS)							
		7*	6	5	4	3	2	1	0
00	0	0	0	1	1	0	0	0	0
01	1	0	0	1	1	0	0	0	1
02	2	0	0	1	1	0	0	1	0
03	3	0	0	1	1	0	0	1	1
04	4	0	0	1	1	0	1	0	0
05	5	0	0	1	1	0	1	0	1
06	6	0	0	1	1	0	1	1	0
07	7	0	0	1	1	0	1	1	1
10	8	0	0	1	1	1	0	0	0
11	9	0	0	1	1	1	0	0	1
12	:	0	0	1	1	1	0	1	0
13	=	0	0	1	1	1	1	0	1
14	'	0	0	1	0	0	1	1	1
15	&	0	0	1	0	0	1	1	0
16	%	0	0	1	0	0	1	0	1
17	[	0	1	0	1	1	0	1	1
20	+	0	0	1	0	1	0	1	1
21	A	0	1	0	0	0	0	0	1
22	B	0	1	0	0	0	0	1	0
23	C	0	1	0	0	0	0	1	1
24	D	0	1	0	0	0	1	0	0
25	E	0	1	0	0	0	1	0	1
26	F	0	1	0	0	0	1	1	0
27	G	0	1	0	0	0	1	1	1
30	H	0	1	0	0	1	0	0	0
31	I	0	1	0	0	1	0	0	1
32	<	0	0	1	1	1	1	0	0
33	.	0	0	1	0	1	1	1	0
34	)	0	0	1	0	1	0	0	1
35	∧	0	1	0	1	1	1	1	0
36	"	0	0	1	0	0	0	1	0
37	;	0	0	1	1	1	0	1	1

*ASCII bit 7 is unassigned and "0" for all codes.

BCD/ASCII CONVERSION TABLE (Cont'd)

6-BIT BCD CODE	8-BIT ASCII CHARACTER	BINARY STATUS OF ASCII CHARACTER (BIT POSITIONS)							
		7*	6	5	4	3	2	1	0
40	-	0	0	1	0	1	1	0	1
41	J	0	1	0	0	1	0	1	0
42	K	0	1	0	0	1	0	1	1
43	L	0	1	0	0	1	1	0	0
44	M	0	1	0	0	1	1	0	1
45	N	0	1	0	0	1	1	1	0
46	O	0	1	0	0	1	1	1	1
47	P	0	1	0	1	0	0	0	0
50	Q	0	1	0	1	0	0	0	1
51	R	0	1	0	1	0	0	1	0
52	!	0	0	1	0	0	0	0	1
53	$	0	0	1	0	0	1	0	0
54	*	0	0	1	0	1	0	1	0
55	#	0	0	1	0	0	0	1	1
56	\	0	1	0	0	0	0	0	0
57	>	0	0	1	1	1	1	1	0
60	Blank	0	0	1	0	0	0	0	0
61	/	0	0	1	0	1	1	1	1
62	S	0	1	0	1	0	0	1	1
63	T	0	1	0	1	0	1	0	0
64	U	0	1	0	1	0	1	0	1
65	V	0	1	0	1	0	1	1	0
66	W	0	1	0	1	0	1	1	1
67	X	0	1	0	1	1	0	0	0
70	Y	0	1	0	1	1	0	0	1
71	Z	0	1	0	1	1	0	1	0
72	]	0	1	0	1	1	1	0	1
73	Comma	0	0	1	0	1	1	0	0
74	(	0	0	1	0	1	0	0	0
75	~	0	1	0	1	1	1	0	0
76	_	0	1	0	1	1	1	1	1
77	?	0	0	1	1	1	1	1	1

*ASCII bit 7 is unassigned and "0" for all codes.

APPENDIX B

SUPPLEMENTARY ARITHMETIC INFORMATION

B. SUPPLEMENTARY ARITHMETIC INFORMATION

NUMBER SYSTEMS

Any number system may be defined by two characteristics, the radix or base and the modulus. The radix or base is the number of unique symbols used in the system. The decimal system has ten symbols, 0 through 9. Modulus is the number of unique quantities or magnitudes a given system can distinguish. For example, an adding machine with ten digits, or counting wheels, would have a modulus of $10^{10}-1$. The decimal system has no modulus because an infinite number of digits can be written, but the adding machine has a modulus because the highest number which can be expressed is 9, 999, 999, 999.

Most number systems are positional; that is, the relative position of a symbol determines its magnitude. In the decimal system, a 5 in the units column represents a different quantity than a 5 in the tens column. Quantities equal to or greater than 1 may be represented by using the 10 symbols as coefficients of ascending powers of the base 10. The number 984_{10} is:

$$\begin{aligned} 9 \times 10^2 &= 9 \times 100 = 900 \\ +8 \times 10^1 &= 8 \times 10 = 80 \\ +4 \times 10^0 &= 4 \times 1 = 4 \\ & \qquad\qquad\qquad 984_{10} \end{aligned}$$

Quantities less than 1 may be represented by using the 10 symbols as coefficients of ascending negative powers of the base 10. The number 0.593_{10} may be represented as:

$$\begin{aligned} 5 \times 10^{-1} &= 5 \times .1 = .5 \\ +9 \times 10^{-2} &= 9 \times .01 = .09 \\ +3 \times 10^{-3} &= 3 \times .001 = .003 \\ & \qquad\qquad\qquad 0.593_{10} \end{aligned}$$

Binary Number System

Computers operate faster and more efficiently by using the binary number system. There are only two symbols, 0 and 1; the base = 2. The following shows the positional value:

...	2^5	2^4	2^3	2^2	2^1	2^0	
	32	16	8	4	2	1	Binary point

The binary number 0 1 1 0 1 0 represents:

$$\begin{aligned} 0 \times 2^5 &= 0 \times 32 = 0 \\ +1 \times 2^4 &= 1 \times 16 = 16 \\ +1 \times 2^3 &= 1 \times 8 = 8 \\ +0 \times 2^2 &= 0 \times 4 = 0 \\ +1 \times 2^1 &= 1 \times 2 = 2 \\ +0 \times 2^0 &= 0 \times 1 = 0 \\ & \qquad\qquad 26_{10} \end{aligned}$$

Fractional binary numbers may be represented by using the symbols as coefficients of ascending negative powers of the base.

	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5} ...
Binary Point	1/2	1/4	1/8	1/16	1/32

The binary number 0. 10 110 may be represented as:

$$\begin{aligned} 1 \times 2^{-1} &= 1 \times 1/2 = 1/2 = 8/16 \\ +0 \times 2^{-2} &= 0 \times 1/4 = 0 = 0 \\ +1 \times 2^{-3} &= 1 \times 1/8 = 1/8 = 2/16 \\ +1 \times 2^{-4} &= 1 \times 1/16 = 1/16 = 1/16 \\ & \qquad\qquad 11/16_{10} \end{aligned}$$

Octal Number System

The octal number system uses eight discrete symbols, 0 through 7. With base eight the positional value is:

...	8^5	8^4	8^3	8^2	8^1	8^0
	32,768	4,096	512	64	8	1

The octal number 513_8 represents:

$$\begin{aligned} 5 \times 8^2 &= 5 \times 64 = 320 \\ +1 \times 8^1 &= 1 \times 8 = 8 \\ +3 \times 8^0 &= 3 \times 1 = 3 \\ & \qquad\qquad 331_{10} \end{aligned}$$

Fractional octal numbers may be represented by using the symbols as coefficients of ascending negative powers of the base.

8^{-1}	8^{-2}	8^{-3}	8^{-4} ...
1/8	1/64	1/512	1/4096

The octal number 0. 4520 represents:

$$\begin{aligned} 4 \times 8^{-1} &= 4 \times 1/8 = 256/512 \\ +5 \times 8^{-2} &= 5 \times 1/64 = 40/512 \\ +2 \times 8^{-3} &= 2 \times 1/512 = 2/512 \\ & \qquad\qquad 298/512 = 149/256_{10} \end{aligned}$$

ARITHMETIC

Addition and Subtraction

Binary numbers are added according to the following rules:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 with a carry of 1

The addition of two binary numbers proceeds as follows (the decimal equivalents verify the result):

Augend	0111	(7)
Addend	+0100	+(4)
Partial Sum	0011	
Carry	1	
Sum	1011	(11)

Subtraction may be performed as an addition:

8 (minuend)		8 (minuend)
-6 (subtrahend)	or	+4 (10's complement of subtrahend)
2 (difference)		2 (difference - omit carry)

The second method shows subtraction performed by the "adding the complement" method. The omission of the carry in the illustration has the effect of reducing the result by 10.

One's Complement: The computer performs all arithmetic and counting operations in the binary one's complement mode. In this system, positive numbers are represented by the binary equivalent and negative numbers in ones' complement notation.

The one's complement representation of a number is found by subtracting each bit of the number from 1. For example:

1111	
-1001	9
0110	(one's complement of 9)

This representation of a negative binary quantity may also be obtained by substituting "1's" for "0's" and "0's" for "1's".

The value zero can be represented in one's complement notation in two ways:

$0000 \rightarrow 00_2$	Positive (+) Zero
$1111 \rightarrow 11_2$	Negative (-) Zero

The rules regarding the use of these two forms for computation are:

- Both positive and negative zero are acceptable as arithmetic operands.
- If the result of an arithmetic operation is zero, it will be expressed as positive zero.

One's complement notation applies not only to arithmetic operations performed in A, but also to the modification of execution addresses in the F register. During address modification, the modified address will equal 77777_8 only if the unmodified execution address equals 77777_8 and b = 0 or $(B^b) = 77777_8$.

Multiplication

Binary multiplication proceeds according to the following rules:

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

Multiplication is always performed on a bit-by-bit basis. Carries do not result from multiplication, since the product of any two bits is always a single bit.

Decimal example:

Multiplicand	14	
Multiplier	12	
Partial Products	28	
	14	(shifted one place left)
Product	168_{10}	

The shift of the second partial product is a shorthand method for writing the true value 140.

Binary example:

Multiplicand (14)	1110	
Multiplier (12)	1100	
	0000	
	0000	shift to place
Partial Products	1110	digits in proper
	1110	columns
Product (168_{10})	10101000_2	

The computer determines the running subtotal of the partial products. Rather than shifting the partial product to the left to position it correctly, the computer right shifts the summation of the partial products one place before the next addition is made. When the multiplier bit is "1", the multiplicand is added to the running total and the results are shifted to the right one place. When the multiplier bit is "0", the partial product subtotal is shifted to the right (in effect, the quantity has been multiplied by 10_2).

Division

The following examples shows the familiar method of decimal division:

```
                  14        Quotient
Divisor    13 ⌐185          Dividend
              13
               55           Partial Dividend
              52
                3           Remainder
```

The computer performs division in a similar manner (using binary equivalents):

```
                    1110     Quotient (14)
Divisor    1101 ⌐10111001    Dividend
                  1101
                 10100
                  1101
                  1110       Partial Dividends
                  1101
                    11       Remainder (3)
```

However, instead of shifting the divisor right to position it for subtraction from the partial dividend (shown above), the computer shifts the partial dividend left, accomplishing the same purpose and permitting the arithmetic to be performed in the A register. The computer counts the number of shifts, which is the number of quotient digits to be obtained; after the correct number of counts, the routine is terminated.

CONVERSIONS

The procedures that may be used when converting from one number system to another are power addition, radix arithmetic, and substitution.

TABLE B-1. RECOMMENDED CONVERSION PROCEDURES (INTEGER AND FRACTIONAL)

Conversion	Recommended Method
Binary to Decimal	Power Addition
Octal to Decimal	Power Addition
Decimal to Binary	Radix Arithmetic
Decimal to Octal	Radix Arithmetic
Binary to Octal	Substitution
Octal to Binary	Substitution
GENERAL RULES	
$r_i > r_f$: use Radix Arithmetic, Substitution $r_i < r_f$: use Power Addition, Substitution r_i = Radix of initial system r_f = Radix of final system	

Power Addition

To convert a number from r_i to r_f ($r_i < r_f$) write the number in its expanded r_i polynomial form and simplify using r_f arithmetic.

EXAMPLE 1 Binary to Decimal (Integer)

$$\begin{aligned} 010\ 111_2 &= 1(2^4) + 0(2^3) + 1(2^2) + 1(2^1) + 1(2^0) \\ &= 1(16) + 0(8) + 1(4) + 1(2) + 1(1) \\ &= 16 + 0 + 4 + 2 + 1 \\ &= 23_{10} \end{aligned}$$

EXAMPLE 2 Binary to Decimal (Fractional)

$$\begin{aligned} .0101_2 &= 0(2^{-1}) + 1(2^{-2}) + 0(2^{-3}) + 1(2^{-4}) \\ &= 0 + 1/4 + 0 + 1/16 \\ &= 5/16_{10} \end{aligned}$$

EXAMPLE 3 Octal to Decimal (Integer)

$$\begin{aligned} 324_8 &= 3(8^2) + 2(8^1) + 4(8^0) \\ &= 3(64) + 2(8) + 4(1) \\ &= 192 + 16 + 4 \\ &= 212_{10} \end{aligned}$$

EXAMPLE 4 Octal to Decimal (Fractional)

$$\begin{aligned} .44_8 &= 4(8^{-1}) + 4(8^{-2}) \\ &= 4/8 + 4/64 \\ &= 36/64_{10} \\ &= 9/16_{10} \end{aligned}$$

Radix Arithmetic

To convert a whole number from r_i to r_f ($r_i > r_f$):

- Divide r_i by r_f using r_i arithmetic
- The remainder is the lowest order bit in the new expression
- Divide the integral part from the previous operation by r_f
- The remainder is the next higher order bit in the new expression
- The process continues until the division produces only a remainder which will be the highest order bit in the r_f expression.

To convert a fractional number from r_i to r_f:

- Multiply r_i by r_f using r_i arithmetic
- The integral part is the highest order bit in the new expression
- Multiply the fractional part from the previous operation by r_f
- The integral part is the next lower order bit in the new expression
- The process continues until sufficient precision is achieved or the process terminates.

EXAMPLE 1 Decimal to Binary (Integer)

45 ÷ 2 = 22 remainder 1; record 1
22 ÷ 2 = 11 remainder 0; record 0
11 ÷ 2 = 5 remainder 1; record 1
5 ÷ 2 = 2 remainder 1; record 1
2 ÷ 2 = 1 remainder 0; record 0
1 ÷ 2 = 0 remainder 1; record 1

Thus: $45_{10} = 101101_2$ 101101

EXAMPLE 2 Decimal to Binary (Fractional)

.25 x 2 = 0.5; record 0
.5 x 2 = 1.0; record 1
.0 x 2 = 0.0; record 0

Thus: $.25_{10} = .010_2$ .010

EXAMPLE 3 Decimal to Octal (Integer)

273 ÷8 = 34 remainder 1; record 1
34 ÷8 = 4 remainder 2; record 2
4 ÷8 = 0 remainder 4; record 4

Thus: $273_{10} = 421_8$ 421

EXAMPLE 4 Decimal to Octal (Fractional)

.55 x 8 = 4.4; record 4
.4 x 8 = 3.2; record 3
.2 x 8 = 1.6; record 1
-- -- -
-- -- -

.431 ...

Thus: $.55_{10} = .431...8$

Substitution

This method permits easy conversion between octal and binary representations of a number. If a number in binary notation is partitioned into triplets to the right and left of the binary point, each triplet may be converted into an octal digit. Similarly, each octal digit may be converted into a triplet of binary digits.

EXAMPLE 1 Binary to Octal

Binary = 110 000 . 001 010
Octal = 6 0 . 1 2

EXAMPLE 2 Octal to Binary

Octal = 6 5 0 . 2 2 7
Binary = 110 101 000 . 010 010 111

FIXED POINT ARITHMETIC

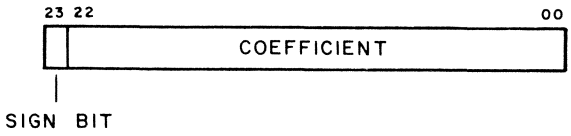
24-Bit Precision

Any number may be expressed in the form kB^n, where 'k' is a coefficient, 'B' a base number, and the exponent 'n' the power to which the base number is raised.

A fixed point number assumes:

- The exponent n = 0 for all fixed point numbers.
- The coefficient, k, occupies the same bit positions within the computer word for all fixed point numbers.
- The radix (binary) point remains fixed with respect to one end of the expression.

A fixed point number consists of a sign bit and coefficient as shown below. The upper bit of any fixed point number designates the sign of the coefficient (23 lower order bits). If the bit is "1", the quantity is negative since negative numbers are represented in one's complement notation; a "0" sign bit signifies a positive coefficient.



The radix (binary) point is assumed to be immediately to the right of the lowest order bit (00).

In many instances, the values in a fixed point operation may be too large or too small to be expressed by the computer. The programmer must position the numbers within the word format so they can be represented with sufficient precision. The process, called scaling, consists of shifting the values a predetermined number of places. The numbers must be positioned far enough to the right in the register to prevent overflow but far enough to the left to maintain precision. The scale factor (number of places shifted) is expressed as the power of the base. For example, $5,100,000_{10}$ may be expressed as 0.51×10^7, 0.051×10^8, 0.0051×10^9, etc. The scale factors are 7, 8, and 9.

Since only the coefficient is used by the computer, the programmer is responsible for remembering the scale factors. Also, the possibility of an overflow during intermediate operations must be considered. For example, if two fractions in fixed point format are multiplied, the result is a number < 1. If the same two fractions are added, subtracted, or divided, the result may be greater than one and an overflow will occur. Similarly, if two integers are multiplied, divided, subtracted or added, the likelihood of an overflow is apparent.

48-Bit Precision (Double Precision)

The 48-bit Add, Subtract, Multiply and Divide instructions enable operands to be processed. The Multiply and Divide instructions utilize the E register and

therefore are executed as trapped instructions if the applicable arithmetic option is not present in a system. Figure 6-5 in the Instruction Section illustrates the operand formats in 48-bit precision Multiply and Divide instructions.

FLOATING POINT ARITHMETIC

As an alternative to fixed point operation a method involving a variable radix point, called floating point, is used. This significantly reduces the amount of bookkeeping required on the part of the programmer.

By shifting the radix point and increasing or decreasing the value of the exponent, widely varying quantities which do not exceed the capacity of the machine may be handled.

Floating point numbers within the computer are represented in a form similar to that used in scientific notation, that is, a coefficient or fraction multiplied by a number raised to a power. Since the computer uses only binary numbers, the numbers are multiplied by powers of two.

$F \cdot 2^E$ where: F = fraction
E = exponent

In floating point, different coefficients need not relate to the same power of the base as they do in fixed point format. Therefore, the construction of a floating point number includes not only the coefficient but also the exponent.

NOTE

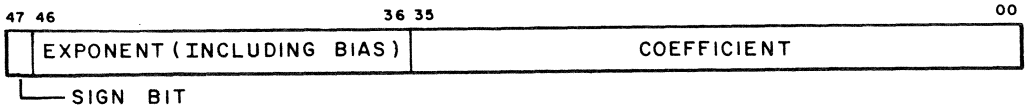
Refer to Figure 6-6 in the Instruction Section for the operand format and bit functions for specific floating point instructions.

Coefficient

The coefficient consists of a 36-bit fraction in the 36 lower order positions of the floating point word. The coefficient is a normalized fraction; it is equal to or greater than 1/2 but less than 1. The highest order bit position (47) is occupied by the sign bit of the coefficient. If the sign bit is a "0", the coefficient is positive; a "1" bit denotes a negative fraction (negative fractions are represented in one's complement notation).

Exponent

The floating point exponent is expressed as an 11-bit quantity with a value ranging from 0000 to 3777_8. It is formed by adding a true positive exponent and a bias of 2000_8 or a true negative exponent and a bias of 1777_8. This results in a range of biased exponents as shown on the following page.



True Positive Exponent	Biased Exponent	True Negative Exponent	Biased Exponent
+0	2000	-0	2000*
+1	2001	-1	1776
+2	2002	-2	1775
--	----	--	----
--	----	--	----
+1776	3776	-1776	0001
$+1777_8$	3777_8	-1777_8	0000_8

*Minus zero is sensed as positive zero by the computer and is therefore biased by 2000_8 rather than 1777_8.

The exponent is biased so that floating point operands can be compared with each other in the normal fixed point mode.

As an example, compare the unbiased exponents of $+52_8$ and $+0.02_8$ (Example 1).

EXAMPLE 1

Number = $+52_8$

0 0 000 000 110 (36 bits)

0 Coefficient Sign — Exponent — Coefficient

Number = $+0.02_8$

1 1 111 111 011 (36 bits)

0 Coefficient Sign — Exponent — Coefficient

In this case +0.02 appears to be larger than +52 because of the larger exponent. If, however, both exponents are biased (Example 2), changing the sign of both exponents makes +52 greater than +0.02.

EXAMPLE 2

Number = $+52_8$

1 0 000 000 110 (36 bits)

0 Coefficient Sign — Exponent — Coefficient

Number = $+0.02_8$

0 1 111 111 011 (36 bits)

0 Coefficient Sign — Exponent — Coefficient

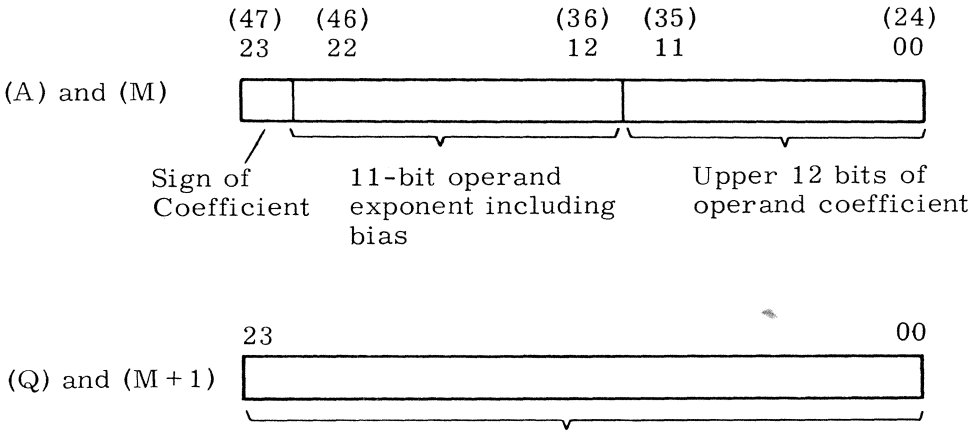
When bias is used with the exponent, floating point operation is more versatile since floating point operands can be compared with each other in the normal fixed point mode.

All floating point operations involve the A, Q, and E registers, plus two consecutive storage locations M and M + 1. The A and Q registers are treated as one 48-bit register. Indirect addressing and address modification are applicable to this whole group of instructions.

Operand Formats

The AQ register and the storage address contents have identical formats.

In both cases the maximum possible shift is 64 (77_8) bit positions. Since the coefficient consists of only 36 bits at the start, any shift greater than 36 positions will, of course, always result in an answer equal to the larger of the two original operands.



Exponents

The 3100, 3200, 3300, 3500 Computers use an 11-bit exponent that is biased by 2000_8 for floating point operations. The effective modulus of the exponent is $\pm 1777_8$ or $\pm 1023_{10}$.

Exponent Equalization

During floating point addition and subtraction, the exponents involved are equalized prior to the operation.

- Addition - The coefficient of the algebraically smaller exponent is automatically shifted right in AQE until the exponents are equal. A maximum of 77_8 shifts may occur.

- Subtraction - If AQ contains the algebraically smaller exponent, the coefficient in AQ is shifted right in AQE until the exponents are equal. If (M) and (M + 1) have the smaller exponent, the complement of the coefficient of (M) and (M + 1) is shifted right in AQE until the exponents are equal or until a maximum of 77_8 shifts are performed.

Rounding

Rounding is an automatic floating point operation and is particularly necessary when floating point arithmetic operations yield coefficient answers in excess of 36 bits.

Although standard floating point format requires only a 36-bit coefficient, portions of the E register are used for extended coefficients. Refer to individual instruction descriptions for E register applications.

Rounding modifies the coefficient result of a floating point operation by adding or subtracting a "1" from the lowest bit position in Q without regard to the biased exponent. The coefficient of the answer in AQ passes through the adder with the rounding quantity before normalization. The conditions for rounding are classified according to arithmetic operation in Table B-2.

TABLE B-2. ROUNDED CONDITIONS FOLLOWING ARITHMETIC OPERATION

Arithmetic OPERATION	Bit 23 of the A Register	Bit 47 of the E Register or (Ratio of Residue/Divisor for Divide Only)	Applicable Rounding
ADD or SUBTRACT	0*	0	No
	0*	1	Add "1"
	1*	0	Subtract "1"
	1*	1	No
	Comments: Rounding occurs as a result of inequality between the sign bits of AQ and E.		
MULTIPLY	0	0	No
	0	1	Add "1"
	1	0	Subtract "1"
	1	1	No
	Comments: A floating point multiplication yields a 76 bit coefficient. Comparison between the sign bits of AQ and E indicates that the lower 36 bits are equal to or greater than 1/2 of the lowest order bit in AQ.		
DIVIDE	0	$\geq$ 1/2 (absolute)	Add "1"
	0	$\leq$ 1/2 (absolute)	No
	1	$\geq$ 1/2 (absolute)	Subtract "1"
	1	$\leq$ 1/2 (absolute)	No
	Comments: Rounding occurs if the answer resulting from the final residue division is equal to or greater than 1/2.		
*Condition of bit 23 of the A register immediately after equalization. (Refer to Exponent Equalization on preceeding page).			

Normalizing

Normalizing brings the above answer back to a fraction with a value between one-half and one with the binary point to the left of the 36th bit of the coefficient. In other words, the final normalized coefficient in AQ will range in value from 2^{36} to $2^{37}-1$ including sign. Arithmetic control normalizes the answer by right or left shifting the coefficient the necessary number of places and adjusting the exponent. It does not shift the residue that is in E.

Faults

Three conditions are considered faults during the execution of floating point instructions:

- Exponent overflow ($> + 1777_8$)
- Exponent underflow ($< - 1777_8$)
- Division by zero, by too small a number, or by a number that is not in floating point format

These faults have several things in common:

- They can be sensed by the INS (77.3) instruction
- Sensing automatically clears them
- The program should sense for these faults only after the floating point instructions have had sufficient time to go to completion
- They may be used to cause an interrupt

FIXED POINT/FLOATING POINT CONVERSIONS

Fixed Point To Floating Point

- Express the number in binary.
- Normalize the number. A normalized number has the most significant 1 positioned immediately to the right of the binary point and is expressed in the range $1/2 \leq k < 1$.
- Inspect the sign of the true exponent. If the sign is positive add 2000_8 (bias) to the true exponent of the normalized number. If the sign is negative, add the bias 1777_8 to the true exponent of the normalized number. In either case, the resulting exponent is the biased exponent.
- Assemble the number in floating point.
- Inspect the sign of the coefficient. If negative, complement the assembled floating point number to obtain the true floating point representation of the number. If the sign of the coefficient is positive, the assembled floating point number is the true representation.

EXAMPLE 1 Convert +4.0 to floating point

- The number is expressed in octal.
- Normalize $4.0 = 4.0 \times 8^0 = 0.100 \times 2^3$
- Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 3.
- Assemble number in floating point format.
  Coefficient = 400 000 000 000_8
  Biased Exponent = 2003_8
  Assembled word = 2003 400 000 000 000_8
- Since the sign of the coefficient is positive, the floating point representation of +4.0 is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

EXAMPLE 2 Convert -4.0 to floating point

- The number is expressed in octal.
- Normalize $-4.0 = -4.0 \times 8^0 = -0.100 \times 2^3$
- Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 3.
- Assemble number in floating point format.
  Coefficient = 400 000 000 000_8
  Biased Exponent = 2003_8
  Assembled word = 2003 400 000 000 000_8
- Since the sign of the coefficient is negative, the assembled floating point word must be complemented. Therefore, the true floating point representation for
  $-4.0 = 5774\ 377\ 777\ 777\ 777_8$.

EXAMPLE 3 Convert 0.5_{10} to floating point

- Convert to octal $0.5_{10} = 0.4_8$
- Normalize $0.4 = 0.4 \times 8^0 = 0.100 \times 2^0$
- Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 0.
- Assemble number in floating point format.
  Coefficient = 400 000 000 000_8
  Biased Exponent = 2000_8
  Assembled word = 2000 400 000 000 000_8
- Since the sign of the coefficient is positive, the floating point representation of $+0.5_{10}$ is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word. This example is a special case of floating point since the exponent of the normalized number is 0 and could be represented as -0. The exponent would then be biased by 1777_8 instead of 2000_8 because of the negative exponent. The 3100 and 3200, however, recognize -0 as +0 and bias the exponent by 2000_8.

EXAMPLE 4 Convert 0.04_8 to floating point

- The number is expressed in octal.
- Normalize $0.04 = 0.04 \times 8^0 = 0.4 \times 8^{-1} = 0.100 \times 2^{-3}$
- Since the sign of the true exponent is negative, add 1777_8 (bias) to the true exponent. Biased exponent = $1777_8 + (-3) = 1774_8$
- Assemble number in floating point format.
  Coefficient = $400\ 000\ 000\ 000_8$
  Biased Exponent = 1774_8
  Assembled word = $1774\ 400\ 000\ 000\ 000_8$
- Since the sign of the coefficient is positive, the floating point representation of 0.04_8 is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

Floating Point to Fixed Point Format

- If the floating point number is negative, complement the entire floating point word and record the fact that the quantity is negative. The exponent is now in a true biased form.
- If the biased exponent is equal to or greater than 2000_8, subtract 2000_8 to obtain the true exponent; if less than 2000_8, subtract 1777_8 to obtain true exponent.
- Separate the coefficient and exponent. If the true exponent is negative, the binary point should be moved to the left the number of bit positions indicated by the true exponent. If the true exponent is positive, the binary point should be moved to the right the number of bit positions indicated by the true exponent.
- The coefficient has now been converted to fixed binary. The sign of the coefficient will be negative if the floating point number was complemented in step one. (The sign bit must be extended if the quantity is placed in a register.)
- Represent the fixed binary number in fixed octal notation.

EXAMPLE 1 Convert floating point number $2003\ 400\ 000\ 000\ 000_8$ to fixed octal

- The floating point number is positive and remains uncomplemented.
- The biased exponent $> 2000_8$; therefore, subtract 2000_8 from the biased exponent to obtain the true exponent of the number. $2003 - 2000 = +3$.
- Coefficient = $400\ 000\ 000\ 000_8 = .100_2$. Move binary point to the right three places. Coefficient = 100.0_2.
- The sign of the coefficient is positive because the floating point number was not complemented in step one.
- Represent in fixed octal notation.
  $100.0 \times 2^0 = 4.0 \times 8^0$.

EXAMPLE 2 Convert floating point number $5774\ 377\ 777\ 777\ 777_8$ to fixed octal

- The sign of the coefficient is negative; therefore, complement the floating point number.
  Complement = $2003\ 400\ 000\ 000\ 000_8$
- The biased exponent (in complemented form) $> 2000_8$; therefore, subtract 2000_8 from the biased exponent to obtain the true exponent of the number 2003 - 2000 = +3.
- Coefficient = $4000\ 000\ 000\ 000_8 = 0.100_2$. Move binary point to the right three places. Coefficient = 100.0_2.
- The sign of the coefficient will be negative because the floating point number was originally complemented.
- Convert to fixed octal. $-100.0_2 = 4.0_8$.

EXAMPLE 3 Convert floating point number $1774\ 400\ 000\ 000\ 000_8$ to fixed octal

- The floating point number is positive and remains uncomplemented.
- The biased exponent $< 2000_8$; therefore, subtract 1777_8 from the biased exponent to obtain the true exponent of the number. $1774_8 - 1777_8 = -3$.
- Coefficient = $400\ 000\ 000\ 000_8 = .100_2$. Move binary point to the left three places. Coefficient = $.000100_2$.
- The sign of the coefficient is positive because the floating point number was not complemented in step one.
- Represent in fixed octal notation.
  $.000100_2 = .04_8$.

APPENDIX C

INSTRUCTION EXECUTION TIMES

C. INSTRUCTION EXECUTION TIMES

The instruction execution times are listed in Table C-1.

Execution time for a given instruction is dependent upon several factors. In determining the times listed, the following criteria were used:

1. A memory cycle time of 0.900 usec.
2. A cable length from the processor to memory of approximately 8 feet.
3. The time for retrieving the instruction from storage (RNI cycle) is included in the listed time; this time also includes time for one indexing operation.
4. Indirect addressing (RADR) was not specified for any instruction.
5. No relocation was performed.
6. Priority conflicts between Main Control and Block Control were prevented.
7. Execution time is decreased on instructions that can be programmed to use memory overlap, i.e., those in which instruction and operand references can use different 16K units of storage. Two times are listed for these instructions.

Execution times for instructions that use indirect addressing are increased by 0.850 usec for each level of indirect addressing performed (this includes time for any required indexing that may be intermixed with the indirect addressing). Relocation time of 0.100 usec is added to execution time only if the current instruction reference is in a different quarter page of memory than the previous instruction reference, or if the current operand reference is in a different quarter page than the previous operand reference. See Section 5 for a description of the optional address relocation feature of the Relocation section.

Due to the asynchronous timing used in the 3500, execution times may vary slightly from processor to processor. The listed times are therefore nominal values. Refer to the parameters and notes following the table for explanatory information.

TABLE C-1. INSTRUCTION EXECUTION TIMES

OCTAL CODE	MNEMONIC CODE	EXECUTION TIME (usec)
00.0	HLT	0.83
00.1-00.6	SJ1-6	(1) 0.83, 0.92
00.7	RTJ	(1) 1.62, 1.73
01	UJP	(1) 0.85, 0.92
02.1-3	IJI	(1) 0.85, 0.92
02.4-7	IJD	(1) 0.85, 0.92
03.0-3	AZJ	1.10
03.4-7	AQJ	1.10
04.0-3	ISE	(1) 1.06, 1.08
04.4-7	ASE, QSE	(1) 1.14, 1.16
05.0-3	ISG	(1) 1.06, 1.08
05.4-7	ASG, QSG	(1) 1.14, 1.16
06	MEQ	(1) (2) 2.05, 2.30
07	MTH	(1) (2) 2.05, 2.30
10.0	SSH	(1) 2.31, 2.81
10.1-3	ISI	1.08
10.4-7	ISD	1.08
11	ECHA	0.75
12.0-3	SHA	0.75
12.4-7	SHQ	0.75
13.0-3	SHAQ	0.75
13.4-7	SCAQ	0.86
14	ENI, ENA, ENQ	0.75
15	INI, INA, INQ, CRI	0.75
16	XOI, XOA, XOQ CRIM, SRIM, SIBA, FRI, DRIP, ERIP, SAD, DAD	0.75
17	ANI, ANA, ANQ	0.75
17.00, 17.04	RST, RSTI	(9) 0.95, 2.95
20	LDA	(1) 1.40, 1.76
21	LDQ	(1) 1.40, 1.76
22	LACH	(1) (3) 1.38, 1.84
23	LQCH	(1) (3) 1.38, 1.84
24	LCA	(1) 1.40, 1.76

TABLE C-1. INSTRUCTION EXECUTION TIMES (Cont'd)

OCTAL CODE	MNEMONIC CODE	EXECUTION TIME (usec)
25	LDAQ	① 2.00, 2.50
26	LCAQ	① 2.00, 2.50
27	LDL	① 1.40, 1.76
30	ADA	① 1.40, 1.76
31	SBA	① 1.40, 1.76
32	ADAQ	① 2.00, 2.50
33	SBAQ	① 2.00, 2.50
34	RAD	① 2.36, 2.76
35	SSA	① 1.40, 1.76
36	SCA	① 1.40, 1.76
37	LPA	① 1.40, 1.76
40	STA	① 1.46, 1.82
41	STQ	① 1.46, 1.82
42	SACH	① ③ 1.44, 1.90
43	SQCH	① ③ 1.44, 1.90
44	SWA	① 1.46, 1.82
45	STAQ	① 2.12, 2.56
46	SCHA	① 1.46, 1.82
47	STI	① 1.44, 1.80
50	MUA	① 3.05, 3.10
51	DVA	① 4.60, 4.70
52	CPR	① 2.00, 2.05
53.(0-7)0	TIA, TAI	0.75
53.01-2	TMQ, TMA	1.08
53.41-2	TQM, TAM	0.96
53.(1-3)3	TMI	1.08
53.(5-7)3	TIM	0.96
53.(0-3)4	AQA, AIA	0.75
53.(5-7)4	IAI	1.36
54	LDI	① 1.40, 1.76
55	RIS	0.75
56	MUAQ	① 5.20, 5.25

TABLE C-1. INSTRUCTION EXECUTION TIMES (Cont'd)

OCTAL CODE	MNMEONIC CODE	EXECUTION TIME (usec)
57	DVAQ	① 8.70, 8.75
60	FAD	① 2.65, 2.75
61	FSB	① 2.65, 2.75
62	FMU	① 4.70, 4.80
63	FDV	① 7.60, 7.70
64.0	MVE	4.00 +0.58 (S)
64.1	MVBF	④ 4.00 +0.58 ($\$_1$) + 0.26 ($\$_2$-$\$_1$)
64.2	MVZF	④ 6.00 +0.58 ($\$_1$) + 0.26 ($\$_2$-$\$_1$)
64.3	MVZS	4.00 +0.58 (S)
64.4	FRMT	3.70 +0.58 ($\$_2$)
64.4	EDIT	⑤ 5.00 +0.60 ($\$_2$)
65	SCAN	3.40 +0.34 ($\$_2$)
66.0	CVDB	5.53 +0.44 ($\$_2$)
66.1	CVBD	6.32 +0.23(M+8N_1+12N_2 +16N_3 + 20N_4 + 24N_5 + 28N_6)
66.2	DTA	4.00 +0.65 ($\$_2$)
66.3	ATD	4.00 +0.65 ($\$_2$)
66.4	PAK	4.00 +0.42 ($\$_2$)
66.5	UPAK	4.00 +0.42 ($\$_2$)
67.0 & 67.1	ADM & SBM	⑤ 5.00 +0.64 (S)
67.2	ZADM	4.00 +0.58 ($\$_1$)+0.26 ($\$_2$ -$\$_1$)
67.3	CMP	4.07 +0.58 (T)
67.4	TST	4.00 +0.28 (L_3)
67.4	TSTN	4.00 +0.28 ($\$_1$)
70.0-2	JMP, HI, ZRO, LOW	0.83, 0.92
70.6	LBR	1.35, 1.70
70.7	SBR	1.41, 1.76
71	SRCE, SRCN	① ⑥ 2.50, 2.85
72	MOVE	① ⑥ 2.50, 2.85
73	INPC	⑥ 1.95
74	INPW	⑥ 1.95

TABLE C-1. INSTRUCTION EXECUTION TIMES (Cont'd)

OCTAL CODE	MNEMONIC CODE	EXECUTION TIME (usec)
73.(1)	INAC	⑥ ⑦ 2.16
74.(1)	INAW	⑥ ⑦ 2.16
75	OUTC	① ⑥ 2.50, 2.85
76	OUTW	① ⑥ 2.50, 2.85
75.(1)	OTAC	⑥ 2.16
76.(1)	OTAW	⑥ 2.16
77.0	CON	⑧ 4.00 to approx 101 usec
77.1	SEL	⑧ 2.00 to approx 101 usec
77.2	COPY	0.93
77.2	EXS	0.93
77.3	CINS	0.93
77.3	INS	0.93
77.4	INTS	0.93
77.50	INCL	0.75
77.510	IOCL	1.02
77.511	CILO	0.90
77.512	CLCA	1.02
77.52	SSIM	0.75
77.53	SCIM	0.75
77.54	ACI	0.75
77.55	CIA	0.75
77.56	JAA	0.75
77.57	IAPR	0.75
77.60	PAUS	1.18 to approx 40 ms
77.61	PRP	1.18 to approx 40 ms
77.61	TMAV	1.65
77.62	SBJP	0.75
77.624	SDL	0.75
77.63	CRA	0.75
77.634	ACR	0.75
77.64	APF	1.18

TABLE C-1. INSTRUCTION EXECUTION TIMES (Cont'd)

OCTAL CODE	MNEMONIC CODE	EXECUTION TIME (usec)
77.65	PFA	1.08
77.66	AOS	0.75
77.664	AIS	0.75
77.67	OSA	0.75
77.674	ISA	0.75
77.70	SLS	0.75
77.71	SFPF	0.75
77.72	SBCD	0.75
77.73	DINT	0.75
77.74	EINT	0.75
77.75	CTI	0.75
77.76	CTO	0.75
77.77	UCS	0.75

① The first time listed was determined using memory overlap; the second time includes no overlap.

② Add 1.2 usec for each location searched other than the first.

③ For instructions using character address indexing, the first time listed does not include indexing, whereas the second time does.

④ The quantity 0.24 ($\$_2$-$\$_1$) is added only if $\$_2 > \$_1$.

⑤ The time listed is for the condition which requires no second pass. For the Edit instruction, add 4.20 + 0.49 ($\$_2$) usec if a second pass is required; for the ADM and SBM instructions, add 3.20 + 0.49 ($\$_2$) usec if a second pass is required.

⑥ For instructions 71-76, execution time is 1.67 usec if the buffer (Search/Move or I/O control) is initially busy.

⑦ Execution time for Input to A instructions is dependent upon external equipment response time. The time listed was determined using an immediate response.

⑧ Execution time for CON and SEL instructions is 0.85 usec if the channel is initially busy.

⑨ The first time listed is for Real-Time interrupt lines 20_8-77_8 which require no restoration; the second time is for lines 00-17_8 which require restoration.

TABLE C-1. INSTRUCTION EXECUTION TIMES (Cont'd)

$\S_1$	=	Number of characters in field A
$\S_2$	=	Number of characters in field C
L_3	=	Number of characters, from right to left, that are of zero value.
M	=	Number of most significant 4-bit binary groups which have a zero value.
N_1	=	Number of 4-bit binary groups to the right of any significant zero groups, up to and including 2 in number. N_1 = 2 if there are 2 or more groups.
N_2	=	1 if there are 3 binary groups, and ='s 2 if there are 4 or more groups.
N_3	=	1 if there are 5 binary groups, and ='s 2 if there are 6 or more groups.
N_4	=	1 if there are 7 binary groups, and ='s 2 if there are 8 or more groups.
N_5	=	1 if there are 9 binary groups, and ='s 2 if there are 10 or more groups.
N_6	=	1 if there are 11 binary groups, and ='s 2 if there are 12 groups.
S	=	Number of characters in the shorter of the two fields $\S_1$ and $\S_2$.
T	=	Number of characters in the longer of the two fields $\S_1$ and $\S_2$.

INSTRUCTION TABLES

TABLE 1. OCTAL LISTING OF INSTRUCTIONS

OCTAL CODE	MNE-MONIC CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
00.0	HLT	m	Unconditional stop, RNI at m on restart	6-12
00.1	SJ1	m	Jump to m if jump key 1 is set	6-17
00.2	SJ2	m	Jump to m if jump key 2 is set	6-17
00.3	SJ3	m	Jump to m if jump key 3 is set	6-17
00.4	SJ4	m	Jump to m if jump key 4 is set	6-17
00.5	SJ5	m	Jump to m if jump key 5 is set	6-17
00.6	SJ6	m	Jump to m if jump key 6 is set	6-17
00.7	RTJ	m	Store P + 1 at m, RNI at m + 1, return to m for P + 1	6-18
01	UJP, I	m, b	Unconditional jump to M	6-19
02.0	NOP		No operation	
02.1-3	IJI	m, b	Index jump to M, incremental	6-20
02.4	NOP		No operation	
02.5-7	IJD	m, b	Index jump to M, decremental	6-21
03.0	AZJ, EQ	m	Jump to m if (A) = 0	6-22
03.1	AZJ, NE	m	Jump to m if (A) $\neq$ 0	6-22
03.2	AZJ, GE	m	Jump to m if (A) $\geq$ 0	6-22
03.3	AZJ, LT	m	Jump to m if (A) $<$ 0	6-22
03.4	AQJ, EQ	m	Jump to m if (A) = (Q)	6-23
03.5	AQJ, NE	m	Jump to m if (A) $\neq$ (Q)	6-23
03.6	AQJ, GE	m	Jump to m if (A) $\geq$ (Q)	6-23
03.7	AQJ, LT	m	Jump to m if (A) $<$ (Q)	6-23
04.0	ISE	y	Skip next instruction if y = 0	6-13
04.1-3	ISE	y, b	Skip next instruction if (B^b) = y	6-13

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (Cont'd)

OCTAL CODE	MNE-MONIC CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
04.4	ASE, S	y	Skip next instruction if (A) = y, sign ext	6-14
04.5	QSE, S	y	Skip next instruction if (Q) = y, sign ext	6-14
04.6	ASE	y	Skip next instruction if (A) = y	6-13
04.7	QSE	y	Skip next instruction if (Q) = y	6-14
05.0	ISG	y	Skip next instruction if y = 0	6-14
05.1-3	ISG	y, b	Skip next instruction if $(B^b) \geq y$	6-14
05.4	ASG, S	y	Skip next instruction if (A) $\geq$ y, sign ext	6-15
05.5	QSG, S	y	Skip next instruction if (Q) $\geq$ y, sign ext	6-15
05.6	ASG	y	Skip next instruction if (A) $\geq$ y	6-15
05.7	QSG	y	Skip next instruction if (Q) $\geq$ y	6-15
06	MEQ	m, i	Masked equality search, skip next instruction if (A) = (Q) AND (M)	6-56
07	MTH	m, i	Masked threshold search, skip next instruction if (A) $\geq$ (Q) AND (M)	6-58
10.0	SSH	m	Shift (m) left one place, end around, skip next instruction if sign negative	6-33
10.1-3	ISI	y, b	Index skip incremental, clear B^b and skip next instruction if (B^b) = y	6-16
10.4-7	ISD	y, b	Index skip decremental, clear B^b and skip next instruction if (B^b) = y	6-16
11.0-3	ECHA	z	Enter character address z into A	6-25
11.4-7	ECHA, S	z	Enter character address z into A, sign ext	6-25

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (Cont'd)

OCTAL CODE	MNE-MONIC CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
12.0-3	SHA	k, b	Shift A, shift count K = k + (B^b)	6-33
12.4-7	SHQ	k, b	Shift Q, shift count K = k + (B^b)	6-35
13.0-3	SHAQ	k, b	Shift AQ, shift count K = k + (B^b)	6-35
13.4-7	SCAQ	k, b	Shift AQ left (end around) until upper 2 bits unequal	6-35
14.0	NOP		No operation	
14.1-3	ENI	y, b	Enter B^b with y	6-25
14.4	ENA, S	y	Enter A with y, sign ext	6-26
14.5	ENQ, S	y	Enter Q with y, sign ext	6-26
14.6	ENA	y	Enter A with y	6-26
14.7	ENQ	y	Enter Q with y	6-26
15.0	CRI		Clear RIP	6-77
15.1-3	INI	y, b	Increase (B^b) by y, sign ext on y and B^b	6-26
15.4	INA, S	y	Increase (A) by y, sign ext	6-27
15.5	INQ, S	y	Increase (Q) by y, sign ext	6-27
15.6	INA	y	Increase (A) by y	6-27
15.7	INQ	y	Increase (Q) by y	6-27
16.00	CRIM	g, x	Selectively clear real-time interrupt mask	6-77
16.01	SRIM	g, x	Selectively set real-time interrupt mask	6-78
16.02	SIBA	d	Set interrupt base address	6-78
16.03	FRI	g, 1	Force real-time interrupt	6-79
16.04	DRIP		Disable RIP	6-79
16.05	ERIP		Enable RIP	6-79
16.06	SAD		Select automatic disable	6-80
16.07	DAD		Deselect automatic disable	6-80
16.1-3	XOI	y, b	Exclusive OR of B^b and y	6-51
16.4	XOA, S	y	Exclusive OR of A and y, sign ext	6-52
16.5	XOQ, S	y	Exclusive OR of Q and y, sign ext	6-52
16.6	XOA	y	Exclusive OR of A and y	6-52
16.7	XOQ	y	Exclusive OR of Q and y	6-52

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (Cont'd)

OCTAL CODE	MNE-MONIC CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
17.00	RST	1	Restore	6-80
17.04	RSTI	1	Restore and Interrupt	6-81
17.1-3	ANI	y, b	AND of B^b and y	6-52
17.4	ANA, S	y	AND of A and y, sign ext	6-53
17.5	ANQ, S	y	AND of Q and y, sign ext	6-53
17.6	ANA	y	AND of A and y	6-53
17.7	ANQ	y	AND of Q and y	6-53
20	LDA, I	m, b	Load A with (M)	6-36
21	LDQ, I	m, b	Load Q with (M)	6-36
22	LACH	r, 1	Load A with character from (R), (B^1) used for indexing	6-37
23	LQCH	r, 2	Load Q with character from (R), (B^2) used for indexing	6-37
24	LCA, I	m, b	Load A with complement of (M)	6-38
25	LDAQ, I	m, b	Load AQ with (M, M + 1)	6-38
26	LCAQ, I	m, b	Load AQ with complement of (M, M + 1)	6-38
27	LDL, I	m, b	Load A with AND of (M) and (Q)	6-53
30	ADA, I	m, b	Add (M) to (A)	6-44
31	SBA, I	m, b	Subtract (M) from (A)	6-44
32	ADAQ, I	m, b	Add (M, M + 1) to (AQ)	6-45
33	SBAQ, I	m, b	Subtract (M, M + 1) from (AQ)	6-45
34	RAD, I	m, b	Add (M) to (A) and replace in M	6-45
35	SSA, I	m, b	Set bits in A for all "1" bits in M	6-54
36	SCA, I	m, b	Complement bits in A for all "1" bits in M	6-54
37	LPA, I	m, b	AND of (M) and (A)	6-55
40	STA, I	m, b	Store (A) in M	6-40
41	STQ, I	m b	Store (Q) in M	6-40

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (Cont'd)

OCTAL CODE	MNE-MONIC CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
42	SACH	r, 2	Store A character in R, use (B^2) to index	6-41
43	SQCH	r, 1	Store Q character in R, use (B^1) to index	6-41
44	SWA, I	m, b	Store word address in M	6-42
45	STAQ, I	m, b	Store (AQ) in M, M + 1	6-42
46	SCHA, I	m, b	Store character address in M	6-42
47	STI, I	m, b	Store (B^b) in M	6-43
50	MUA, I	m, b	Multiply (A) by (M), product appears in QA	6-46
51	DVA, I	m, b	Divide (AQ) by (M), quotient appears in A, remainder in Q	6-46
52	CPR, I	m, b	1) If (M)> (A), RNI at P + 1 2) If (Q) > (M), RNI at P + 2 3) If (A) > (M) $\geq$ (Q), RNI at P + 3	6-64
53.(0-3)0	TIA	b	Transfer (B^b) to A	6-28
53.40	NOP		No operation	
53.(5-7)0	TAI	b	Transfer (A) to B^b	6-29
53.(0-3)1	TMQ	v	Transfer (register v) to Q	6-29
53.(4-7)1	TQM	v	Transfer (Q) to register v	6-29
53.(0-3)2	TMA	v	Transfer (register v) to A	6-29
53.(4-7)2	TAM	v	Transfer (A) to register v	6-30
53.03	NOP		No operation	
53.(1-3)3	TMI	v, b	Transfer (register v) to B^b	6-30
53.43	TIM	v	Clear register v	6-30

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (Cont'd)

OCTAL CODE	MNE-MONIC CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
53.(5-7)3	TIM	v, b	Transfer (B^b) to register v	6-30
53.04	AQA		Transfer (A) + (Q) to A	6-31
53.(1-3)4	AIA	b	Transfer (A) + (B^b) to A	6-31
53.44	NOP		No operation	
53.(5-7)4	IAI	b	Transfer (A) + (B^b) to B^b, sign of B^b ext	6-31
54	LDI, I	m, b	Load B^b with (M_{00-14})	6-39
55.0	RIS		Relocate to instruction state	6-82
55.1	ELQ		Transfer (E_L) to Q	6-31
55.2	EUA		Transfer (E_U) to A	6-32
55.3	EAQ		Transfer (E) to AQ	6-32
55.4	ROS		Relocate to operand state	6-83
55.5	QEL		Transfer (Q) to E_L	6-32
55.6	AEU		Transfer (A) to E_U	6-32
55.7	AQE		Transfer (AQ) to E	6-32
56	MUAQ, I	m, b	Multiply (AQ) by (M, M + 1), product appears in AQE	6-46
57	DVAQ, I	m, b	Divide (AQE) by (M, M + 1), quotient appears in AQ, remainder in E	6-47
60	FAD, I	m, b	Floating point: add (M, M + 1) to (AQ)	6-48
61	FSB, I	m, b	Floating point: subtract (M, M + 1) from (AQ)	6-48
62	FMU, I	m, b	Floating point: multiply (AQ) by (M, M + 1), product appears in AQ	6-49
63	FDV, I	m, b	Floating point: divide (AQ) by (M, M + 1), quotient appears in AQ, remainder in E	6-49
64.0	MVE	r, B_r, §1 s, B_s, §2	Move characters from fld A to fld C	6-116
64.0	MVE, dc	r, B_r, s, B_s, §2	Move characters from fld A to fld C with delimiting	6-118

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (Cont'd)

OCTAL CODE	MNE-MONIC CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
64.1	MVBF	r, B_r, §1 s, B_s, §2	Move characters from fld A to fld C with blank fill	6-119
64.2	MVZF	r, B_r, §1 s, B_s, §2	Move characters from fld A to fld C with zero fill	6-120
64.3	MVZS	r, B_r, §1 s, B_s, §2	Move characters from fld A to fld C, suppressing leading zeros	6-121
64.3	MVZS, dc	r, B_r, s, B_s, §2	Move characters from fld A to fld C with de-limiting, suppressing leading zeros	6-122
64.4	EDIT	r, B_r, §1 s, B_s, §2	Perform complete edit according to COBOL picture stored in fld C	6-127
64.4	FRMT	r, B_r, §1 s, B_s, §2	Perform limited editing according to picture stored in fld C	6-125
65.0	SCAN, LR, EQ	r, B_r, §2, sc	Scan fld A, left to right, for equality	6-133
65.0	SCAN LR, EQ dc	r, B_r, §2 sc	Scan fld A, left to right, for equality - delimiting available	6-134
65.1	SCAN RL, EQ	r, B_r, §2, sc	Scan fld A, right to left, for equality	6-137
65.1	SCAN RL, EQ, dc	r, B_r, §2, sc	Scan fld A, right to left, for equality - delimiting available	6-138
65.2	SCAN, LR, NE	r, B_r, §2, sc	Scan fld A, left to right, for inequality	6-135
65.2	SCAN, LR, NE, dc	r, B_r, §2, sc	Scan fld A, left to right, for inequality - delimiting available	6-136
65.3	SCAN, RL, NE	r, B_r, §2, sc	Scan fld A, right to left, for inequality	6-139
65.3	SCAN, RL, NE, dc	r, B_r, §2, sc	Scan fld A, right to left, for inequality - delimiting available	6-140
66.0	CVDB	r, B_r, §1, m, B_m	Convert BCD to binary	6-141

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (Cont'd)

OCTAL CODE	MNE-MONIC CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
66.1	CVBD	m, B_m, n, B_n	Convert binary to BCD	6-142
66.2	DTA	r, B_r, $ 2 m, B_m	Convert BCD to ASCII	6-143
66.2	DTA, dc	r, B_r, $ 2 m, B_m	Convert BCD to ASCII - delimiting available	6-144
66.3	ATD	m, B_m, $ 2, s, B_s	Convert ASCII to BCD	6-145
66.3	ATD, dc	m, B_m, $ 2, s, B_s	Convert ASCII to BCD - delimiting available	6-146
66.4	PAK	r, B_r, $ 2, m, B_m	Pack 6-bit BCD into 4-bit BCD	6-147
66.5	UPAK	m, B_m, s, B_s, $ 2	Unpack 4-bit BCD into 6-bit BCD	6-148
67.0	ADM	r, B_r, $ 1 s, B_s, $ 2	Add fld A to fld C, sum in fld C	6-149
67.1	SBM	r, B_r, $ 1 s, B_s, $ 2	Subtract fld A from fld C, difference in fld C	6-152
67.2	ZADM	r, B_r, $ 1 s, B_s, $ 2	Move characters, right to left, from fld A to fld C with zero fill.	6-123
67.3	CMP	r, B_r, $ 1 s, B_s, $ 2	Compare fld A to fld C, terminate upon unequal comparison	6-153
67.3	CMP, dc	r, B_r, s, B_s, $ 2	Compare fld A to fld C, terminate upon unequal comparison, or delimiting character	6-157
67.4	TST	r, B_r, $ 1	Test fld A for +, -, or 0	6-159
67.4	TSTN	r, B_r, $ 1	Test fld A for numeric	6-160
70.0	JMP, HI	m	Jump if BDP condition register >0 or +	6-24
70.1	JMP, ZRO	m	Jump if BDP condition register = 0	6-24
70.2	JMP, LOW	m	Jump if BDP condition register <0 or -	6-24
70.6	LBR	m	Load BDP conditions from (m)	6-161

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (Cont'd)

OCTAL CODE	MNE-MONIC CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
71	SRCE, INT	sc, r, s	Search fld r for equality to sc	6-59
71	SRCN, INT	sc, r, s	Search fld r for inequality to sc	6-60
72	MOVE INT	§, r, s	Move § characters from r to s	6-62
73. 0-3	INPC, INT, B, H, G	ch, r, s	Character-addressed input to storage	6-98
73. 4-7	INAC, INT	ch	Character input to A	6-101
74. 0-3	INPW, INT, B, N, G	ch, m, n	Word-addressed input to storage	6-99
74. 4-7	INAW, INT	ch	Word input to A	6-102
75. 0-3	OUTC, INT, B, H	ch, r, s	Character-addressed output from storage	6-104
75. 4-7	OTAC, INT	ch	Character output from A	6-107
76. 0-3	OUTW, INT, B, N	ch, m, n	Word-addressed output from storage	6-105
76. 4-7	OTAW, INT	ch	Word output from A	6-108
77. 0	CON	x, ch	Connect	6-92
77. 1	SEL	x, ch	Select function	6-94
77. 2ch, x; x = 0	COPY	ch	External status to A_{00-11}, interrupt mask to A_{12-23}	6-66
77. 2ch, x; x ≠ 0	EXS	x, ch	Sense external status as defined by mask x	6-65
77. 3ch, x; x = 0	CINS	ch	Internal status to A_{00-11}, interrupt mask to A_{12-23}	6-68
77. 3ch, x; x ≠ 0	INS	x, ch	Sense internal status as defined by mask x	6-67
77. 4	INTS	x, ch	Sense interrupt as defined by mask x	6-68
77. 50	INCL	x	Clear interrupts as defined by mask x	6-74

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (Cont'd)

OCTAL CODE	MNE-MONIC CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
77.51	IOCL	x	Clear I/O channel, typewriter, or S/M control	6-91
77.511	CILO	cm	Lock-out interrupts on channel(s) specified by mask cm	6-74
77.512	CLCA	cm	Clear channel(s) specified by mask cm	6-92
77.52	SSIM	x	Set interrupt mask register as defined by x	6-73
77.53	SCIM	x	Clear interrupt mask register as defined by x	6-73
77.54	ACI		Transfer (A_{00-02}) to channel index register	6-90
77.55	CIA		Transfer (channel index register) to A_{00-02}	6-90
77.56	JAA		Transfer last jump instruction address to A	6-83
77.57	IAPR		Interrupt associated processor	6-75
77.60	PAUS	x	Pause if busy lines designated by x are active	6-69
77.61x x ≠ 0	PRP	x	PAUS, with no real-time clock incrementing	6-71
77.61x x = 0	TMAV		Send storage request, RNI at P + 2 if reply received in 5 usec	6-83
77.62	SBJP		Enables transfer from monitor to program state at next jump instruction	6-84
77.624	SDL		Set destructive load A condition in condition register	6-85
77.63	CRA		Transfer (CR) to A_{00-05}	6-85
77.634	ACR		Transfer (A_{00-05}) to CR	6-86

TABLE 1. OCTAL LISTING OF INSTRUCTIONS (Cont'd)

OCTAL CODE	MNE-MONIC CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
77.64	APF	w, 2	Store (A_{00-11}) in page file index 'w', use (B^2) to index	6-86
77.65	PFA	w, 2	Load A_{00-11} with (page file index 'w'), use (B^2) to index	6-86
77.66	AOS		Transfer (A_{00-02}) to operand state register	6-87
77.664	AIS		Transfer (A_{00-02}) to instruction state register	6-87
77.67	OSA		Transfer (operand state register) to A_{00-02}	6-87
77.674	ISA		Transfer (instruction state register) to A_{00-02}	6-87
77.70	SLS		Stop if selective stop switch is set	6-12
77.71	SFPF		Set floating point fault	6-75
77.72	SBCD		Set BCD fault	6-75
77.73	DINT		Disable interrupt control	6-73
77.74	EINT		Enable interrupt control	6-72
77.75	CTI		Set console typewriter input	6-96
77.76	CTO		Set console typewriter output	6-97
77.77	UCS		Unconditional stop, RNI at P + 1 on restart	6-12

TABLE 2. MNEMONIC LISTING OF INSTRUCTIONS

MNE-MONIC CODE	OCTAL CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
ACI	77.54		Transfer (A_{00-02}) to channel index register	6-90
ACR	77.634		Transfer (A_{00-05}) to CR	6-86
ADA, I	30	m, b	Add (M) to (A)	6-44
ADAQ, I	32	m, b	Add (M, M + 1) to (AQ)	6-45
ADM	67.0	r, B_r, $ 1, s, B_s, $ 2	Add fld A to fld C, sum in fld C	6-149
AEU	55.6		Transfer (A) to E_U	6-32
AIA	53.(1-3)4	b	Transfer (A) + (B^b) to A	6-31
AIS	77.664		Transfer (A_{00-02}) to instruction state register	6-87
ANA	17.6	y	AND of A and y	6-53
ANA, S	17.4	y	AND OF A and y, sign ext	6-53
ANI	17.1-3	y, b	AND of B^b and y	6-52
ANQ	17.7	y	AND of Q and y	6-53
ANQ, S	17.5	y	AND of Q and y, sign ext	6-53
AOS	77.66		Transfer (A_{00-02}) to operand state register	6-87
APF	77.64	w, 2	Store (A_{00-11}) in page file index w, use (B^2) to index	6-86
AQA	53.04		Transfer (A) + (Q) to A	6-31
AQE	55.7		Transfer (AQ) to E	6-32
AQJ, EQ	03.4	m	Jump to m if (A) = (Q)	6-23
AQJ, GE	03.6	m	Jump to m if (A) $\geq$ (Q)	6-23
AQJ, LT	03.7	m	Jump to m if (A) < (Q)	6-23
AQJ, NE	03.5	m	Jump to m if (A) $\neq$ (Q)	6-23
ASE	04.6	y	Skip next instruction if (A) = y	6-13
ASE, S	04.4	y	Skip next instruction if (A) = y, sign ext	6-14
ASG	05.6	y	Skip next instruction if (A) $\geq$ y	6-15
ASG, S	05.4	y	Skip next instruction if (A) $\geq$ y, sign ext	6-15

TABLE 2. MNEMONIC LISTING OF INSTRUCTIONS (Cont'd)

MNE-MONIC CODE	OCTAL CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
AZJ, EQ	03.0	m	Jump to m if (A) = 0	6-22
AZJ GE	03.2	m	Jump to m if (A) $\geq$ 0	6-22
AZJ, LT	03.3	m	Jump to m if (A) $<$ 0	6-22
AZJ, NE	03.1	m	Jump to m if (A) $\neq$ 0	6-22
CIA	77.55		Transfer (channel index register) to A_{00-02}	6-90
CILO	77.511	cm	Lockout interrupts on channel(s) specified by mask cm	6-74
CINS	77.3	x, ch	Internal status to A_{00-11}; interrupt mask to A_{12-23}	6-68
CLCA	77.512	cm	Clear channel(s) specified by mask cm	6-92
CMP	67.3	r, B_r, $\$_1$, s, B_s, $\$_2$	Compare fld A to fld C, terminate upon unequal comparison	6-153
CMP dc	67.3	r, B_r, s, B_s, $\$_2$	Compare fld A to fld C, terminate upon unequal comparison, or delimiting character	6-157
CON	77.0	x, ch	Connect	6-92
COPY	77.2	ch	External status to A_{00-11}; interrupt mask to A_{12-23}	6-66
CPR, I	52	m, b	1. If (M) $>$ (A), RNI at P + 1 2. If (Q) $>$ (M), RNI at P + 2 3. If (A) $\geq$ (M) $\geq$ (Q), RNI at P + 3	6-64
CRA	77.63		Transfer (CR) to A_{00-05})	6-85
CRI	15.0		Clear RIP	6-77
CRIM	16.00	g, x	Selectively clear real-time interrupt mask	6-77
CTI	77.75		Set console typewriter input	6-96

TABLE 2. MNEMONIC LISTING OF INSTRUCTIONS (Cont'd)

MNE-MONIC CODE	OCTAL CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
CTO	77.76		Set console typewriter output	6-97
CVBD	66.1	m, B_m, n, B_n	Convert binary to BCD	6-142
CVDB	66.0	r, B_r, §1, m, B_m	Convert BCD to binary	6-141
DAD	16.07		Deselect automatic disable	6-80
DINT	77.73		Disable interrupt control	6-73
DRIP	16.04		Disable RIP	6-79
DVA, I	51	m, b	Divide (AQ) by (M), quotient appears in A, remainder in Q	6-46
DVAQ, I	57	m, b	Divide (AQE) by (M, M + 1), quotient appears in AQ, remainder in E	6-47
EAQ	55.3		Transfer (E) to AQ	6-32
ECHA	11.0-3	z	Enter character address z into A	6-25
ECHA, S	11.4-7	z	Enter character address z, sign ext, into A	6-25
EDIT	64.4	r, B_r, §1, s, B_s, §2	Perform complete edit according to COBOL picture stored in fld C	6-127
EINT	77.74		Enable interrupt control	6-72
ELQ	55.1		Transfer (E_L) to Q	6-31
ENA	14.6	y	Enter A with y	6-26
ENA, S	14.4	y	Enter A with y, sign extended	6-26
ENI	14.1-3	y, b	Enter B^b with y	6-25
ENQ	14.7	y	Enter Q with y	6-26
ENQ, S	14.5	y	Enter Q with y, sign ext	6-26
ERIP	16.05		Enable RIP	6-79
EUA	55.2		Transfer (E_U) to A	6-32

TABLE 2. MNEMONIC LISTING OF INSTRUCTIONS (Cont'd)

MNE-MONIC CODE	OCTAL CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
EXS	77.2	x, ch	Sense external status as defined by mask x	6-65
FAD, I	60	m, b	Floating point: add (M, M + 1) to AQ	6-48
FDV, I	63	m, b	Floating point: divide (AQ) by (M, M + 1), quotient appears in AQ, remainder in E	6-49
FMU, I	62	m, b	Floating point: multiply (AQ) by (M, M + 1), product appears in AQ	6-49
FRI	16.03	g, 1	Force real-time interrupt	6-79
FRMT	64.4	r, B_r, $\$_1$, s, B_s, $\$_2$	Performs limited editing according to picture stored in fld C.	6-125
FSB, I	61	m, b	Floating point: subtract (M, M + 1) from (AQ)	6-48
HLT	00.0	m	Unconditional stop, RNI at m upon restart	6-12
IAI	53.(5-7)4	b	Transfer (A) + (B^b) to B^b, sign of B^b ext	6-31
IAPR	77.57		Interrupt associated processor	6-75
IJD	02.5-7	m, b	Index jump to M, decremental	6-21
IJI	02.1-3	m, b	Index jump to M, incremental	6-20
INA	15.6	y	Increase (A) by y	6-27
INA, S	15.4	y	Increase (A) by y, sign ext	6-27
INAC, INT	73.4-7	ch	Character input to A	6-101
INAW, INT	74.4-7	ch	Word input to A	6-102
INCL	77.50	x	Clear interrupts as defined by mask x	6-74
INI	15.1-3	y, b	Increase (B^b) by y sign ext on y and B^b	6-26
INPC, INT, B, H, G	73.0-3	ch, r, s	Character addressed input to storage	6-98

TABLE 2. MNEMONIC LISTING OF INSTRUCTIONS (Cont'd)

MNE-MONIC CODE	OCTAL CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
INPW, INT, B, N, G	74. 0-3	ch, m, n	Word addressed input to storage	6-99
INQ	15. 7	y	Increase (Q) by y	6-27
INQ, S	15. 5	y	Increase (Q) by y, sign ext	6-27
INS	77. 3	x, ch	Sense internal status as defined by mask x	6-67
INTS	77. 4	x, ch	Sense interrupt as defined by mask x	6-68
IOCL	77. 51	x	Clear I/O channel, typewriter, or S/M control	6-91
ISA	77. 674		Transfer (instruction state register) to A_{00-02}	6-87
ISD	10. 4-7	y, b	Index skip decremental; clear B^b, skip next instruction if $(B^b) = y$	6-16
ISE	04. 0	y	Skip next instruction if y = 0	6-13
ISE	04. 1-3	y, b	Skip next instruction if $(B^b) = y$	6-13
ISG	05. 0	y	Skip next instruction if y = 0	6-14
ISG	05. 1-3	y, b	Skip next instruction if $(B^b) \geq y$	6-14
ISI	10. 1-3	y, b	Index skip incremental; clear B^b, skip next instruction if $(B^b) = y$	6-16
JAA	77. 56		Transfer last jump instruction address to A	6-83
JMP, HI	70. 0	m	Jump if BDP condition register > 0 or +	6-24
JMP, ZRO	70. 1	m	Jump if BDP condition register = 0	6-24
JMP, LOW	70. 2	m	Jump if BDP condition register < 0 or -	6-24
LACH	22	r, 1	Load A with character from (R), use (B^1) to index	6-37

TABLE 2. MNEMONIC LISTING OF INSTRUCTIONS (Cont'd)

MNE-MONIC CODE	OCTAL CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
LBR	70.6	m	Load BDP conditions from (m)	6-161
LCA, I	24	m, b	Load A with complement of (M)	6-38
LCAQ, I	26	m, b	Load AQ with complement of (M, M + 1)	6-38
LDA, I	20	m, b	Load A with (M)	6-36
LDAQ, I	25	m, b	Load AQ with (M, M + 1)	6-38
LDI, I	54	m, b	Load B^b with (M_{00-14})	6-39
LDL, I	27	m, b	Load A with AND of (M) and (Q)	6-53
LDQ, I	21	m, b	Load Q with (M)	6-36
LPA, I	37	m, b	AND of (M) and (A)	6-55
LQCH	23	r, 2	Load Q with character from (R), use (B^2) to index	6-37
MEQ	06	m, i	Masked equality search, skip next instruction if (A) = (Q) AND (M)	6-56
MOVE, INT	72	§, r, s	Move § characters from r to s	6-62
MTH	07	m, i	Masked threshold search, skip next instruction if (A) $\geq$ (Q) AND (M)	6-58
MUA, I	50	m, b	Multiply (A) by (M), product appears in QA	6-46
MUAQ, I	56	m, b	Multiply (AQ) by (M, M + 1), product appears in AQE	6-46
MVBF	64.1	r, B_r, $§_1$, s, B_s, $§_2$	Move characters from fld A to fld C with blank fill	6-119
MVE	64.0	r, B_r, $§_1$, s, B_s, $§_2$	Move characters from fld A to fld C	6-116
MVE, dc	64.0	r, B_r, s, B_s, $§_2$	Move characters from fld A to fld C with delimiting	6-118

TABLE 2. MNEMONIC LISTING OF INSTRUCTIONS (Cont'd)

MNE-MONIC CODE	OCTAL CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
MVZF	64.2	r, B_r, $\S_1$, s, B_s, $\S_2$	Move characters from fld A to fld C with zero fill	6-120
MVZS	64.3	r, B_r, $\S_1$, s, B_s, $\S_2$	Move characters from fld A to fld C, suppressing leading zeros	6-121
MVZS, dc	64.3	r, B_r, s, B_s, $\S_2$	Move characters from fld A to fld C with delimiting, suppressing leading zeros	6-122
OSA	77.67		Transfer (operand state register) to A_{00-02}	6-87
OTAC, INT	75.4-7	ch	Character output from A	6-107
OTAW, INT	76.4-7	ch	Word output from A	6-108
OUTC, INT, B, H	75.0-3	ch, r, s	Character-addressed output from storage	6-104
OUTW, INT, B, N	76, 0-3	ch, m, n	Word-addressed output from storage	6-105
PAK	66.4	r, B_r, $\S_2$, m, B_m	Pack 6-bit BCD into 4-bit BCD	6-147
PAUS	77.60	x	Pause if busy lines defined by x are active	6-69
PFA	77.65	w, 2	Load A_{00-11} with (page file index 'w'), use (B^2) to index	6-86
PRP	77.61	x	PAUS, with no real-time clock incrementing	6-71
QEL	55.5		Transfer (Q) to E_L	6-32
QSE	04.7	y	Skip next instruction if (Q) = y	6-14
QSE, S	04.5	y	Skip next instruction if (Q) = y, sign ext	6-14
QSG	05.7	y	Skip next instruction if (Q) $\geq$ y	6-15
QSG, S	05.5	y	Skip next instruction if (Q) $\geq$ y, sign ext	6-15

TABLE 2. MNEMONIC LISTING OF INSTRUCTIONS (Cont'd)

MNE-MONIC CODE	OCTAL CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
RAD, I	34	m, b	Add (M) to (A) and replace in M	6-45
RIS	55.0		Relocate to instruction state	6-82
ROS	55.4		Relocate to operand state	6-83
RST	17.00	1	Restore	6-80
RSTI	17.04	1	Restore and interrupt	6-81
RTJ	00.7	m	Store P + 1 at m, RNI at m + 1, return to m for P + 1	6-18
SACH	42	r, 2	Store A character in R, use (B^2) to index	6-41
SAD	16.06		Select automatic disable	6-80
SBA, I	31	m, b	Subtract (M) from (A)	6-44
SBAQ, I	33	m, b	Subtract (M, M + 1) from (AQ)	6-45
SBCD	77.72		Set BCD fault	6-75
SBJP	77.62		Enables transfer from monitor to program state at next jump instruction	6-84
SBM	67.1	r, B_r, $\S_1$, s, B_s, $\S_2$	Subtract fld A from fld C, difference in fld C	6-152
SCA, I	36	m, b	Complement bits in A for "1" bits in M	6-54
SCAN, LR, EQ, dc	65.0	r, B_r, $\S_2$, sc	Scan fld A, left to right, for equality - delimiting available	6-134
SCAN, LR, NE, dc	65.2	r, Br, $\S_2$, sc	Scan fld A, left to right, for inequality - delimiting available	6-136
SCAN, RL, EQ, dc	65.1	r, B_r, $\S_2$, sc	Scan fld A, right to left, for equality - delimiting available	6-138
SCAN, RL, NE, dc	65.3	r, B_r, $\S_2$, sc	Scan fld A, right to left, for inequality - delimiting available	6-140

TABLE 2. MNEMONIC LISTING OF INSTRUCTIONS (Cont'd)

MNE-MONIC CODE	OCTAL CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
SCAN, LR, EQ	65.0	r, B_r, $\S_2$, sc	Scan fld A, left to right, for equality	6-133
SCAN, LR, NE	65.2	r, B_r, $\S_2$, sc	Scan fld A, left to right, for inequality	6-135
SCAN, RL, EQ	65.1	r, B_r, $\S_2$, sc	Scan fld A, right to left, for equality	6-137
SCAN, RL, NE	65.3	r, B_r, $\S_2$, sc	Scan fld A, right to left, for inequality	6-139
SCAQ	13.4-7	k, b	Shift AQ left (end around) until upper 2 bits unequal	6-35
SCHA, I	46	m, b	Store character address in M	6-42
SCIM	77.53	x	Selectively clear interrupt mask as defined by x	6-73
SDL	77.624		Set destructive load A condition in condition register	6-85
SEL	77.1	x, ch	Select function	6-94
SFPF	77.71		Set floating point fault	6-75
SHA	12.0-3	k, b	Shift A, shift count $K = k + (B^b)$	6-33
SHAQ	13.0-3	k, b	Shift AQ, shift count $K = k + (B^b)$	6-35
SHQ	12.4-7	k, b	Shift Q, shift count $K = k + (B^b)$	6-35
SIBA	16.02	d	Set interrupt base address	6-78
SJ1	00.1	m	Jump to m if jump key 1 is set	6-17
SJ2	00.2	m	Jump to m if jump key 2 is set	6-17
SJ3	00.3	m	Jump to m if jump key 3 is set	6-17
SJ4	00.4	m	Jump to m if jump key 4 is set	6-17
SJ5	00.5	m	Jump to m if jump key 5 is set	6-17
SJ6	00.6	m	Jump to m if jump key 6 is set	6-17

TABLE 2. MNEMONIC LISTING OF INSTRUCTIONS (Cont'd)

MNE-MONIC CODE	OCTAL CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
SLS	77.70		Stop if selective stop switch is set	6-12
SQCH	43	r, 1	Store Q character in R, use (B^1) to index	6-41
SRCE, INT	71	sc, r, s	Search fld r for equality to sc	6-59
SRCN, INT	71	sc, r, s	Search fld r for in-equality to sc	6-60
SSA, I	35	m, b	Set bits in A for all "1" bits in M	6-54
SSH	10.0	m	Shift left one place, end around, skip next in-struction if sign negative	6-33
SRIM	16.01	g, x	Selectively set real-time interrupt mask	6-78
SSIM	77.52	x	Set Interrupt mask reg-ister as defined by x	6-73
STA, I	40	m, b	Store (A) in M	6-40
STAQ, I	45	m, b	Store (AQ) in M, M + 1	6-42
STI, I	47	m, b	Store (B^b) in M	6-43
STQ, I	41	m, b	Store (Q) in M	6-40
SWA, I	44	m, b	Store word address in M	6-42
TAI	53.(5-7)0	b	Transfer (A) to B^b	6-29
TAM	53.(4-7)2	v	Transfer (A) to register v	6-30
TIA	53.(0-3)0	b	Transfer (B^b) to A	6-28
TIM	53.(5-7)3	v, b	Transfer (B^b) to register v	6-30
TMA	53.(0-3)2	v	Transfer (register v) to A	6-29
TMAV	77.61		Send storage request, RNI at P + 2 if reply received in 5 usec	6-83
TMI	53.(1-3)3	v, b	Transfer (register v) to B^b	6-30
TMQ	53.(0-3)1	v	Transfer (register v) to Q	6-29
TQM	53.(4-7)1	v	Transfer (Q) to register v	6-29

TABLE 2. MNEMONIC LISTING OF INSTRUCTIONS (Cont'd)

MNE-MONIC CODE	OCTAL CODE	AD-DRESS FIELD	OPERATION	PAGE NO.
TST	67.4	r, B_r, $ 1	Test fld A for +, -, or 0	6-159
TSTN	67.4	r, B_r, $ 1	Test fld A for numeric	6-160
UCS	77.77		Unconditional stop, RNI at P + 1 upon restart	6-12
UJP, I	01	m, b	Unconditional jump to M	6-19
UPAK	66.5	m, B_m, s, B_s, $ 2	Unpack 4-bit BCD into 6-bit BCD	6-148
XOA	16.6	y	Exclusive OR of A and y	6-52
XOA, S	16.4	y	Exclusive OR of A and y, sign ext	6-52
XOI	16.1-3	y, b	Exclusive OR of B^b and y	6-51
XOQ	16.7	y	Exclusive OR of Q and y	6-52
XOQ, S	16.5	y	Exclusive OR of Q and y, sign ext	6-52
ZADM	67.2	r, B_r, $ 1, s, B_s, $ 2	Move characters, right to left, from fld A to fld C with zero fill	6-123

INDEX

COMMENT SHEET

MANUAL TITLE CONTROL DATA® 3500 COMPUTER SYSTEM

Reference Manual

PUBLICATION NO. 60200300 REVISION D

FROM: NAME: _____

BUSINESS ADDRESS: _____

COMMENTS:

This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number references and fill in publication revision level as shown by the last entry on the Record of Revision page at the front of the manual. Customer engineers are urged to use the TAR.

CUT ALONG LINE

AA3419 REV. 11/69 PRINTED IN U.S.A.

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Technical Publications Department
4201 North Lexington Avenue
Arden Hills, Minnesota 55112

CUT ALONG LINE

FOLD

FOLD

QUICK REFERENCE INSTRUCTION INDEX

CONTROL DATA
CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO., MINNEAPOLIS, MINN. 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

Litho in U.S.A.