HONEYWELL

HARDWARE

MODEL 8200

HARDWARE REFERENCE MANUAL

SUBJECT: Detailed Information Concerning the Central Processor Hardware of the Model 8200. Also Describes Instruction Repertoire and Scientific Unit, and Provides Summary Information About Peripheral Devices.

SPECIAL INSTRUCTIONS: The information contained herein will be superseded by more complete information at a later date.

DATE: August 1, 1967

FILE NO.: 113.0011.0000.0-685*

0456
1M
1.5969
Printed in U.S.A.

*Underscoring denotes Order Number.

FOREWORD

This manual, which is part of the Model 8200 System Publications Library, constitutes a reference source of detailed information concerning the central processor of the Model 8200. It is assumed that the reader has a basic understanding of the Model 8200 System. For this purpose, the Model 8200 Summary Description (Order No. 191) is suggested preliminary reading.

Section I of this manual introduces the Model 8200 and describes some of its significant features. The second section presents a description of the information formats utilized by the system. Section III describes the memory subsystem while Section IV details the input/output subsystem (including a brief description of associated peripheral devices). Detailed information concerning the word processor and the variable-length-field (VLF) processor can be found in Sections V and VI, respectively. Also included in these sections is the internal machine language for each processor and a description of the manner in which that language is interpreted and manipulated. In addition, Appendices A through G contain supplementary information for both processors.

Detailed information concerning the associated peripheral devices is not included in this manual but, instead, can be found in the appropriate peripheral device manuals. Software information pertaining to the MOD 8 Operating System can be found in other manuals of the Model 8200 System Publications Library.

A hardware programming and operating handbook may be constructed by combining in a single binder this manual and the manuals/bulletins pertaining to the peripheral components of the installed system. This manual and the peripheral device manuals are all published in loose-leaf format for ease of rapid updating by means of replacement-page addenda.

The equipment characteristics reported herein remain subject to change in order to allow the introduction of design improvements.

TABLE OF CONTENTS

TABLE OF CONTENTS (cont)

TABLE OF CONTENTS (cont)

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (cont)

LIST OF TABLES

LIST OF TABLES (cont)

SECTION I
INTRODUCTION

Honeywell's Model 8200 is a powerful extension of Series 200. This computer is designed to handle large-scale business and scientific applications, as well as real-time communication, and multiprogrammed, time-sharing applications. Its ability to deal efficiently with such applications derives from the provision of hardware-controlled multiprogramming, multiprocessing, high computation speeds, large main memory capacity, high-speed input/output devices, and optional floating-point arithmetic. The system is compatible with both the Series 200 and H-800/1800 systems.

A Model 8200 consists of five basic elements (see Figure 1-1): a word-processing subsystem, a variable-length-field (VLF) processing subsystem, a memory subsystem, an input/output subsystem, and a master control facility which coordinates the activities of these elements. Interaction and communication between elements are accomplished by means of control instructions and program interrupts. This section describes these five basic elements, the manner in which they communicate with one another, and the expansion of processing power that is possible through the addition of optional hardware features.

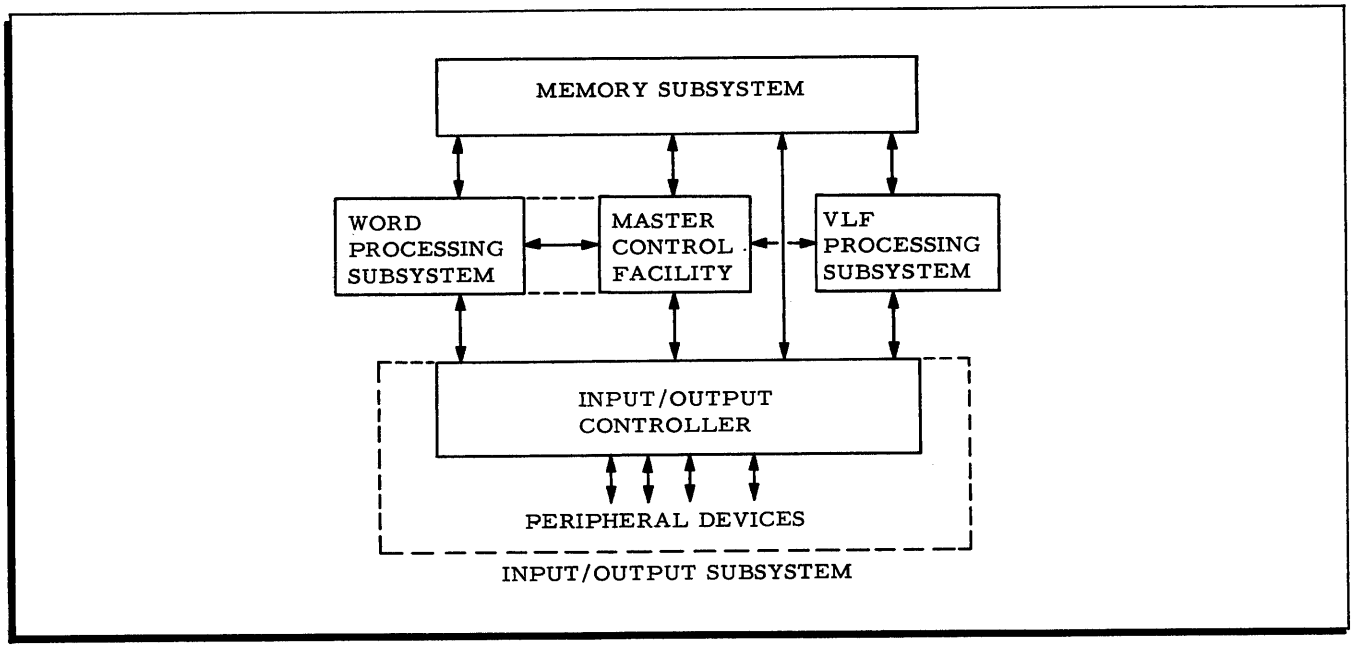


Figure 1-1. Model 8200 System

Word Processing Subsystem

The word processing subsystem executes 3-address instructions which manipulate data in decimal, alphanumeric, binary, and floating-point formats. As many as eight independent programs can be run concurrently at internal speeds averaging 400,000 three-address operations per second. This multiprogramming capability is implemented by the use of eight independent control groups, each consisting of 32 program control registers.

The control registers within a group are 24 bits in length, enabling explicit addressing of all main memory. These registers also provide facilities for indexed, indirect, and indexed indirect addressing of main memory. In addition, two sequencing registers are included which allow dual sequencing of program operations.

Multiprogram control directs the time sharing of the word processor for the active program groups. Each of the programs being processed is assigned one or more of the control groups. Once a program is initiated, its corresponding control group of registers directs the selection of instructions. In this way, each program may start, proceed, and stop independently of all other programs.

As mentioned above, memory locations may be addressed in direct, indirect, or indexed form. Direct addresses refer to specific memory locations or control registers. Indirect addresses use the contents of a control register to locate an operand in memory. Indexed addresses are partial addresses which are added to a base stored in an index register to obtain the complete address. In this way, an entire series of addresses can be modified easily and automatically.

The word processing subsystem also utilizes an extended addressing facility which allows direct addressing of the entire main memory, regardless of its size. Extended addressing is also provided for the other basic forms of addressing: extended indirect and extended indexed addressing of main memory. The extended addressing capability also permits the use of any of the 32 control registers within a group as index registers.

Automatic masking facilities are included as part of the word processing subsystem. Through this masking process, a programmer can compare portions of information words, perform arithmetic operations on portions of words, and select data from within a word for transfer to other locations for processing.

VLF Processing Subsystem

The VLF processing subsystem executes character oriented instructions and processes

data in variable length fields. Data is manipulated in binary and decimal formats. The VLF processor performs arithmetic and logical operations at an average execution rate of 100,000 two-address instructions per second.

Like the word processor, the VLF processor contains a group of program control registers. These registers contain the addresses of instructions and data being processed during the course of a program. Control registers can be addressed either by programmed instruction or from the operators' console.

To facilitate VLF programming, memory may be addressed in direct, indirect, or indexed form. Control register length enables the VLF processor to have direct access to 524, 288 main memory characters. Indirect addressing enables a programmer to refer to stored information by way of one or more intermediary addresses. Index registers provide an automatic means for address modification without altering the referenced instruction. In addition, 2-, 3-, and 4-character address interpretation allows direct addressing of the above mentioned characters, while at the same time saving processing time and memory space when working in localized areas of memory.

Memory Subsystem

The memory subsystem consists of either two, four, or eight modules of core memory and a memory controller. Each module is four characters in width and 32,768 four-character groups in length. The basic memory contains 262,144 characters (32,768 words); this may be expanded to a maximum of 1,048,576 characters (131,072 words). Both characters and words are individually addressable. The main memory cycle time for either a 4-character group or a word is 750 nanoseconds. A parity bit is provided for each 6-bit character (8 parity bits per word) and parity is checked whenever data is read out of memory.

The memory controller is designed to provide maximum simultaneity of memory operations. This is achieved by an interleaved addressing scheme together with the ability to cycle each memory module independently. Up to four accesses to four different modules can occur simultaneously. Thus, word processing, VLF processing, and input/output operations can proceed simultaneously. The memory controller, in addition to handling and routine multiple requests for access, also provides for resolution of conflicting requests and for memory protection control.

The memory subsystem allows protected memory areas to be designated in blocks of 4,096 characters (512 words). Each of the eight word programs, the VLF programs, and the master control facility have unique identification tags which allow access to their associated blocks of protected memory.

Input/Output Subsystem

The input/output subsystem is composed of an input/output (I/O) controller and a wide range of peripheral equipment. The I/O controller provides the path through which peripheral orders are issued and the facilities for controlling the resulting data transfer.

In the basic Model 8200, the I/O controller has two sets of 16 read/write channels. One set performs the read/write functions for the word processor while the other set performs the same functions for the VLF processor. Any combination of 16 read/write channels can be used simultaneously thus, allowing a like number of I/O operations to proceed simultaneously with internal word and VLF processing.

The basic system also has three input/output sectors. Each sector can accommodate the permanent connection of up to 16 peripheral controls. Read/write channels are not permanently associated with a particular sector, but instead are capable of variable assignment to any sector. This arrangement provides for the maximum number of simultaneous I/O operations to take place in each sector.

Data transfers between peripheral devices and main memory via the I/O controller occur at a total combined rate of over 1.3 million characters per second in the basic system. These data transfers are overlapped with processor operations and do not interrupt the processors unless a memory conflict occurs (i.e., simultaneous demand for the same memory module), in which case the memory controller grants priority to the I/O controller.

Any of the Series 200 peripheral equipment may be connected to the I/O controller. A total of 48 peripheral controls are allowed in the basic system; however, by use of multiple device controls, the number of connected devices can be increased substantially. Most H-800/1800 peripheral equipment may be connected to the I/O controller.

Master Control Facility

The master control facility is essentially a ninth word-processing program control group. However, it has much broader responsibilities than the other eight control groups. This facility, in conjunction with the master system control program, maintains the overall coordination of all system activities. The master control hardware and software, referred to as master control, can perform the following functions automatically:

- Set memory partitions and allocate blocks of memory to individual programs.
- Analyze and resolve attempted memory usage violations.
- Assign peripheral devices to active program control groups and enforce peripheral device protection.

- Diagnose and resolve issuance of illegal orders (i.e., undefined op codes), program violations, and when possible, machine malfunctions such as parity errors.
- Provide a communication link between the word processor, VLF processor, and the I/O controller.
- Set or alter various privileged control functions such as base relocation registers, stopper relocation registers, and key registers.
- Provide supervision over programs running in the word and VLF processors.

Master control contains all of the special registers of a word processor control group and can execute all word processor instructions. Also, in order to perform the above functions, master control is equipped with special capabilities. These include a set of privileged instructions which are extensions to the word processor instruction repertoire and which only master control can execute. Included in the other functional units of the system is a master control interrupt facility which allows them to signal problems and to request the services of master control. A master control interrupt is accompanied by the information necessary to identify the calling unit and to ascertain its requirements. Master control is equipped with special instructions which enable it to reply to a calling program's interrupt and also to interrupt the other functional units of the system.

In addition, master control has privileged access to memory partition indicators in the memory controller and to peripheral device protection and reservation indicators in the input/output controller. This ensures that master control can maintain overall coordinating control of memory and peripheral device utilization.

Master control has three modes of operation: ready, hunt, and no hunt. In the ready mode, master control can be activated by any program in either the word or VLF processors, or by the input/output controller. In the hunt mode, master control is effectively the ninth program control group of the word processor. In this mode, master control can be called in the same manner as in the ready mode. When running in the no hunt mode, master control has complete control of the word processor and cannot be called by any other element in the system. When master control is in one of the calling modes (ready or hunt), a call will change its mode of operation to no hunt in order to service the call.

All interrupt conditions in the Model 8200 are recognized and serviced by master control. Table 1-1 lists the five general categories of interrupt conditions.

Table 1-1. Interrupts to the Master Control Facility

CATEGORY OF INTERRUPT	EXTERNAL	MASTER CONTROL	I/O	VLF PROCESSOR	WORD PROCESSOR
PRIORITY	1	2	3	4	5
CAUSE OF INTERRUPT	Receipt of data from communication device	Memory parity Master control fault Service request Add/subtract overflow Divide over-capacity Exponent underflow Exponent overflow	I/O operation fault Peripheral interruption	Machine fault Program fault Master call Peripheral order	Trapped orders Machine fault Program fault Master call Peripheral order Instruction time-out Program no hunt loop*

*An internal timer can be set to one of sixteen stages ranging from 500 nanoseconds to 524 milliseconds.

Interprocessor Communication

An interprocessor communication system is provided in the Model 8200 which enables both processors, master control, and the I/O controller to communicate with one another. This system allows one element to request data or action from, or to provide data or a response to, another element. The following communication paths are possible in the Model 8200.

- Master control to and from the word processor.
- Master control to and from the VLF processor.
- Master control to and from the I/O controller.
- I/O controller to and from the word processor.
- I/O controller to and from the VLF processor.

Communication is implemented by furnishing each element with a buffer area located in main memory. The buffer stores information such as a reason code, a sub-reason code, a group number, and in some instances status information. The reason code identifies the nature of the call. For calls that are processed automatically by hardware, this code is generally the operation or command code of the operation to be performed. Reason codes include the following: peripheral orders issued by a processor to the I/O controller, trapping of specified peripheral orders by master control, program or processor malfunctions to be resolved by master control, and other inquiries and responses enabling master control to maintain overall coordination of all system activities. The sub-reason code gives additional information concerning the reason for the call while the group number identifies the calling element. In some instances, the status of an element is required in order to aid in handling a call (i.e., the status of a processor after a machine malfunction).

When communication takes place, the calling element places the coded information in the buffer area. The called element fetches this information from the buffer, interprets it, and performs the requested function. A response is made to the caller indicating recognition of the call. If the caller requests information to be returned, the called element places this data into the buffer area prior to its response. All interprocessor communication is handled by the system hardware and the master control program.

Console

The console, an integral part of the Model 8200, allows operator communication with a program via instructions entered through the console keyboard and program communication with the operator via console typewriter messages. The console also includes display lights which indicate the status of the various programs that are operating in both the word and VLP processors.

Optional Features

Two features are available which greatly increase the processing power of the Model 8200. These features are:

- Feature 8201-B - Scientific Unit
- Feature 8214 - Expanded I/O Capability

FEATURE 8201-B

This feature provides the Model 8200 with floating-point arithmetic. The scientific unit handles decimal values from 10^{-64} to 10^{+63} and binary values from 16^{-64} to 16^{+63}. The decimal mantissa provides 10-digit precision; 40-place precision is provided by the binary mantissa. Description of this unit and the instructions it provides are presented in Appendix A.

FEATURE 8214

With this feature, an additional set of 16 read/write channels is provided for the word processor, enabling the Model 8200 to perform an additional 18 simultaneous input/output operations. Thus, up to 34 simultaneous input/output operations can be performed on the expanded system. This feature also provides facilities for attaching an additional 48 peripheral controls to the system. In addition, the I/O controller can accommodate a peak data transfer rate of over 2.8 million characters per second. A detailed description of this feature is presented in Section IV.

SECTION II
INFORMATION FORMAT

WORD PROCESSOR FORMAT

The basic unit of information in the word processing subsystem is a fixed-length word consisting of 48 data bits and eight parity bits used by the automatic checking circuitry. A main memory word may represent a machine instruction or one or more pieces of data. In addition to the main memory, the word processor includes a control memory of program control registers, used primarily for control purposes and address modification. A control register has the capacity to store a partial word consisting of up to 24 information bits.

Data Words

A computer program generally manipulates data in one or more different forms: decimal, alphanumeric, binary, or a combination of these. The word processor is capable of handling all these types of information. It may interpret the 48 bits of a word in groups of four for the purpose of binary-coded decimal operation, in groups of six for alphanumeric operation, or as individual units of information for pure binary operation. Figure 2-1 illustrates the structures of these different words.

A decimal word contains either 11 decimal digits with a sign or 12 decimal digits without sign. The decimal arithmetic instructions interpret a word as a sign and 11 digits. The sign consists of four bits which may represent either the sign of the entire word or individual 1-bit signs for as many as four different pieces of information within the word. Although a positive sign is normally represented by four binary ones and a negative sign by four binary zeros, a nonstandard configuration is perfectly acceptable as input to the arithmetic unit, which interprets any combination of bits except four binary zeros as a positive sign. The sign supplied with the result of an arithmetic operation, however, is always one of the two standard conventions, either four binary ones or four binary zeros.

An alphanumeric word is comprised of eight 6-bit groups. Each group can represent any of 26 alphabetic characters, 10 decimal digits, or 20 special characters such as punctuation marks, etc., (see Table 2-1). Numbers may be stored in alphanumeric (6-bit) form, but the arithmetic unit cannot manipulate them as such; it handles numbers in pure binary or binary-coded decimal form.

BIT POSITION	1	5	9	13	17	21	25	29	33	37	41	45
DECIMAL	±	1	2	3	4	5	6	7	8	9	0	1
ALPHANUMERIC	R	O		B	I		N	S		O	N	
ALPHANUMERIC COMPRESSED	C			W	E		B	B		1	7	4
BINARY	±	(44 Binary Digits)										
FLOATING-POINT DECIMAL	±	Exp'nt (7 binary digits)		Mantissa (10 Decimal Digits)								
FLOATING-POINT BINARY	±	Exp'nt (7 binary digits)		Mantissa (40 Binary Digits)								
INSTRUCTION	COMMAND CODE			ADDRESS A			ADDRESS B			ADDRESS C		
CONTROL REGISTER (Normal)							±		(15 Binary Digits)			
CONTROL REGISTER (Extended)							±	(23 Binary Digits)				

Figure 2-1. Model 8200 Word Structure

The 48 binary digits of a word may also represent a pure binary number, which may be stored as a sign and 44 bits, or as 48 unsigned bits. With the exception of the instructions Word Add and Word Difference, which treat their operands as 48-bit unsigned numbers, the binary arithmetic instructions interpret operands as signed 44-bit numbers. The sign convention in binary arithmetic is identical to that described for decimal words.

A word can also be handled as a floating-point number, composed of a 1-bit sign, a 7-bit exponent, and a 40-bit mantissa. The floating-point decimal word has an exponent that can represent a power of 10 from the -64th to the +63rd, and a mantissa that can represent a 10-digit number from .1000000000 through .9999999999 (when normalized). In floating-point binary form (represented in hexadecimal notation), the exponent can represent a power of 16 from the -64th to the +63rd, and the mantissa a 40-bit number from .00010000...0000 through .1111...1111 (when normalized); the mantissa can represent the equivalent of 12 decimal digits. A one in the sign-bit position of the floating-point word indicates that the number is positive, and a zero that it is negative. The floating-point unit is discussed in Appendix A.

Instruction Words

The 48 bits of an instruction word are normally interpreted as four fields of 12 bits each. As shown in Figure 2-1, bits 1 through 12 represent the command code; bits 13 through 24, 25 through 36, and 37 through 48 are designated as the A-address field, B-address field, and C-address field, respectively. The address portions of instructions normally are used to designate the locations of operands and results, but in certain instructions they may contain special information such as the number of words to be moved, the number of bits to be shifted, a change of sequence counter, and so forth.

In addition, the word processor also incorporates an extended instruction format. This extended format occupies two consecutive 48-bit words, and consists of a 24-bit command code and three 24-bit address fields (see Figure 2-2).

In certain instances, the extended instruction format need not occupy two consecutive words of main memory. If two of the three 24-bit address fields in an instruction are not required by the nature of the instruction, then the extended instruction is compressed into one 48-bit word, consisting of a 24-bit command code and one 24-bit address field (see Figure 2-2).

A detailed discussion of the instructions (normal and extended formats) and the types of addressing used can be found in Section V.

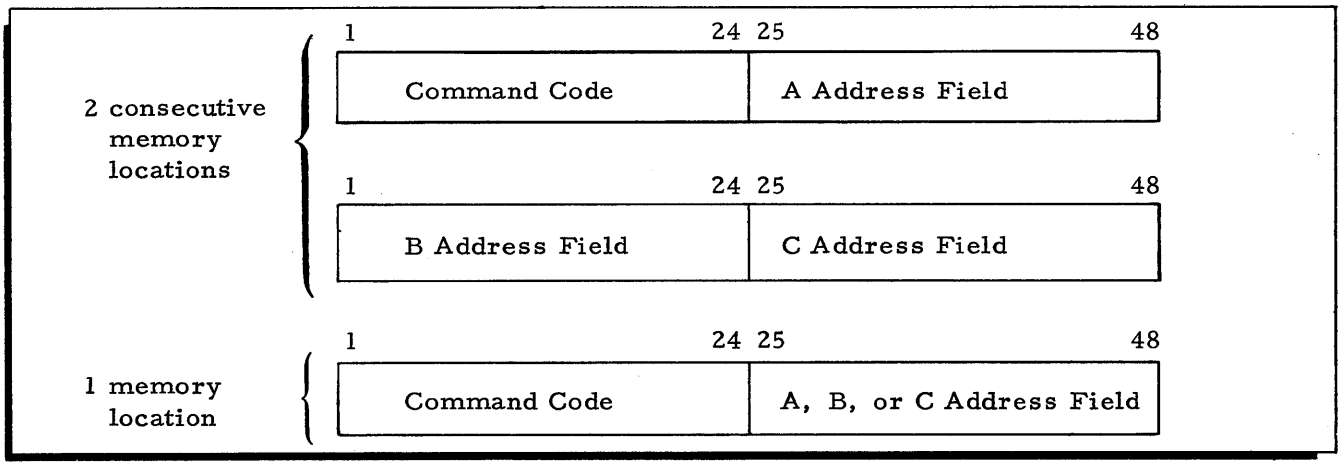


Figure 2-2. Extended Instruction Formats

Control Register Words

As previously noted, a control register can store up to 24 information bits. Normally, when these bits are manipulated within the control register circuitry, the bit position 2^{15} is interpreted as a sign (1 = plus, 0 = minus). However, instructions which consider control registers as 24-bit operands will interpret the 2^{23} bit position as the sign bit. Depending upon the type of addressing used, the remaining 15 or 23 bits of a control register word may be

interpreted as a main memory address, consisting of array and bank indicators and a subaddress, or as a control register address, consisting of a group indicator and a subaddress. When a control register word is modified arithmetically within the control register circuitry, the value of the sign bit determines whether it is incremented or decremented.

Special Words

Two special words — the end-of-record word and the end-of-item word — are used in the word processor.

The end-of-record word is used to designate the end of a group of words constituting a single record. An end-of-record word is automatically generated in the word processor during the execution of a Record Transfer instruction.

As the name implies, an end-of-item word is used to designate the end of a group of words constituting a single item within a record. A record may contain an unspecified number of items, each of which is followed by an end-of-item word, or in the case of the last item, by an end-of-record word. An end-of-item word is automatically generated in the word processor during the execution of an Item Transfer instruction.

The function of both special words will be highlighted in the discussion of the Record and Item Transfer instructions (see Section V).

VLF PROCESSOR FORMAT

The basic unit of information in the VLF processing subsystem is a 9-bit character, consisting of 6 data bits, 2 punctuation bits, and a parity bit used by the automatic checking circuitry.

Information is stored in variable-length groups of adjacent character positions in memory called fields. Each location within a field stores either six binary digits or one alphanumeric character. Since fields can vary in length, information units of varying length can be stored without wasting memory capacity.

Consecutive fields can be combined to form larger units of information called an item. Grouping fields to form an item simplifies the manipulation of related data fields and minimizes the number of instruction executions required to move consecutive fields within main memory.

Any unit of information that is to be transferred between main memory and a peripheral device is defined as a record. Records, like fields, can be of any length.

The use of a variable-length data format requires that there be a method of indicating the actual length of a unit of information. This requirement is fulfilled by the two punctuation bits associated with each character location. These bits constitute a word mark — used to define the length of a field; an item mark — used to define the length of an item; or a record mark — used to define the length of a record. Figure 2-3 provides a summary of data storage unit characteristics.

FORMAT	DEFINITION	LENGTH DEFINED BY
FIELD	A group of consecutive characters which are treated as a unit.	Word Mark
ITEM	One or more consecutive fields.	Item Mark
RECORD	A unit of information to be transferred between a peripheral device and the main memory.	Record Mark

Figure 2-3. Data Storage Characteristics

In addition to defining the lengths of data fields, word marks are also used to define the lengths of instructions in memory.

Instruction Format

Instructions, like data units, are variable in length. The basic instruction format consists of an operation code (op code) which specifies the type of operation to be performed, two operand fields which specify the binary addresses of fields to be used in the operation, and a variant character(s).

The variant character is used to expand the meaning of the op code or to specify literally a piece of data to be used in the operation. However, there are many times when not all of these instruction elements are needed, in which case they may be omitted to minimize both the amount of memory storage required and the time necessary to retrieve and execute an instruction. The most common instruction formats are illustrated in Figure 2-4.

PERIPHERAL DEVICE FORMATS

Magnetic Tape

In many applications, a major input and output medium for the Model 8200 is magnetic tape. The standard Model 8200 uses 1/2-inch tapes as the recording medium. A tape system using 3/4-inch tape is also available.

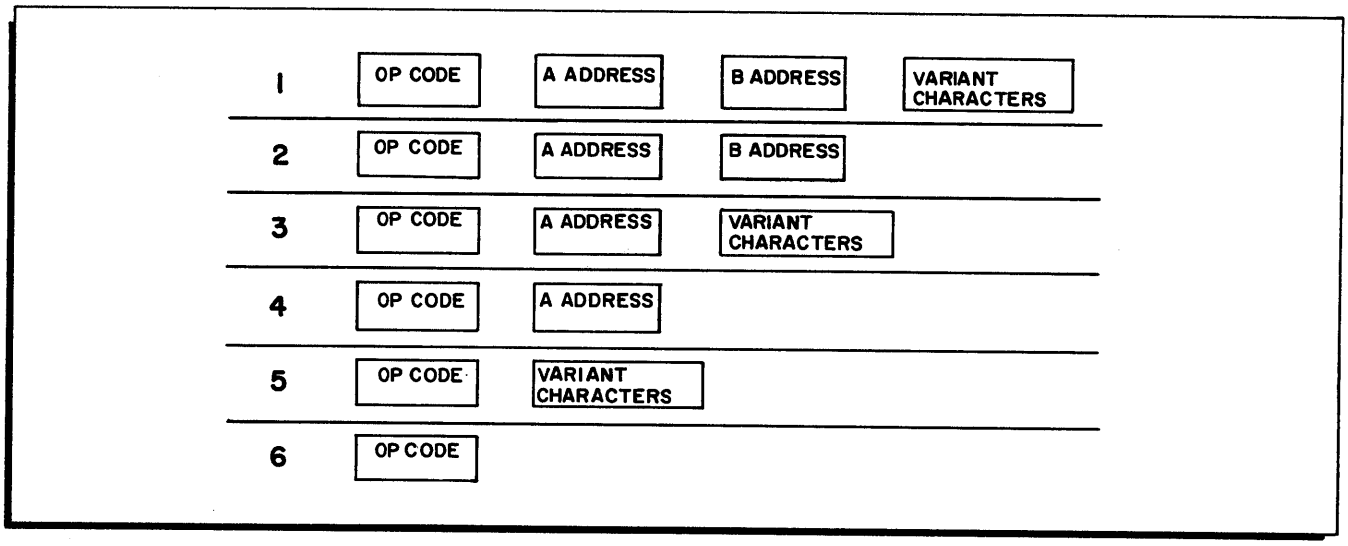


Figure 2-4. VLF Processor Instruction Formats

Information is stored on 1/2-inch magnetic tape in variable-length groups of characters called records. The tape is divided lengthwise into seven recording channels. A line of bit positions across the tape, one position for each channel, is called a frame. The seven bits in a frame consist of six information bits and one parity bit. Notice that no channels are provided for the storage of punctuation bits on tape. Unlike main memory records, which are delimited by record-mark punctuation, tape records are separated from each other by a band of blank tape, which is called an interrecord gap. The representation of a memory character position on magnetic tape is shown in Figure 2-5.

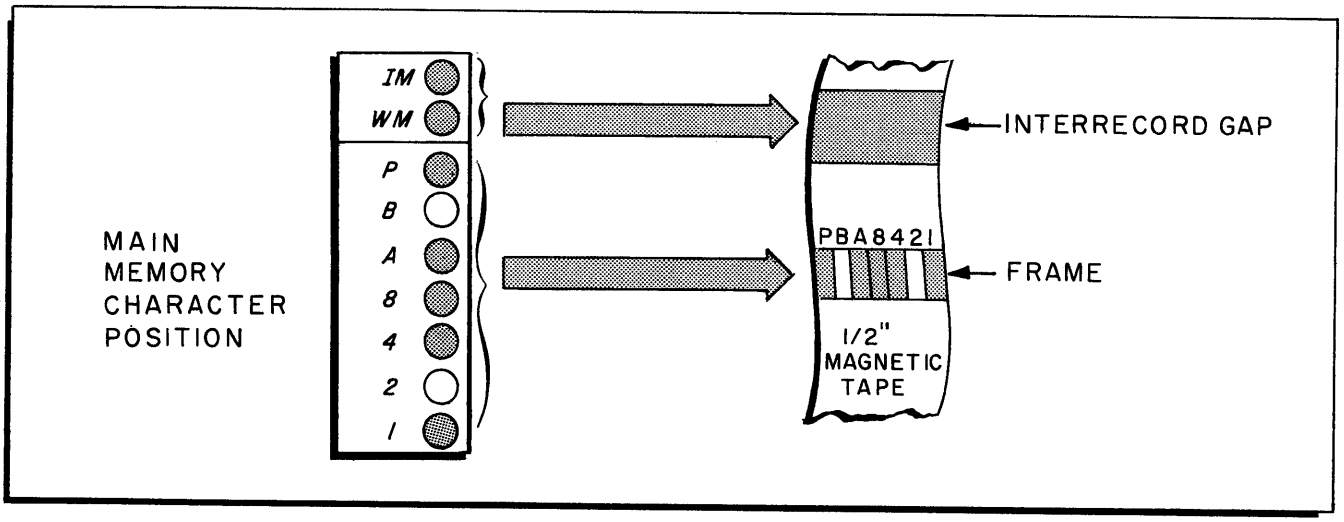


Figure 2-5. Character Representation on Magnetic Tape

Characters recorded on magnetic tape are transferred from main memory without parity bits. At the time of recording, the magnetic tape control generates parity bits as required. The programmer may specify either odd- or even-parity recording; in the odd-parity mode the bit count in each frame is odd; in the even-parity mode the bit count is even.

In addition to parity bits, which are used for frame checking, the magnetic tape control also generates a longitudinal check frame which is used for channel checking purposes. A check frame is automatically appended at the end of each record stored on tape.

Recall that a record stored in memory is delimited by a record mark in the character position following the last character in the record. When a record is transferred to tape, the contents of the character position containing the record mark are not included as part of the record. On the other hand, if a record mark is sensed in memory when information is being read in from tape, the record mark will terminate the record and the character position containing the record mark will receive a character from the tape. Although data transfer from the tape is terminated by the record mark, tape motion continues until an interrecord gap is sensed. No punctuation marks are altered in any way as a result of tape read/write operations initiated by a program.

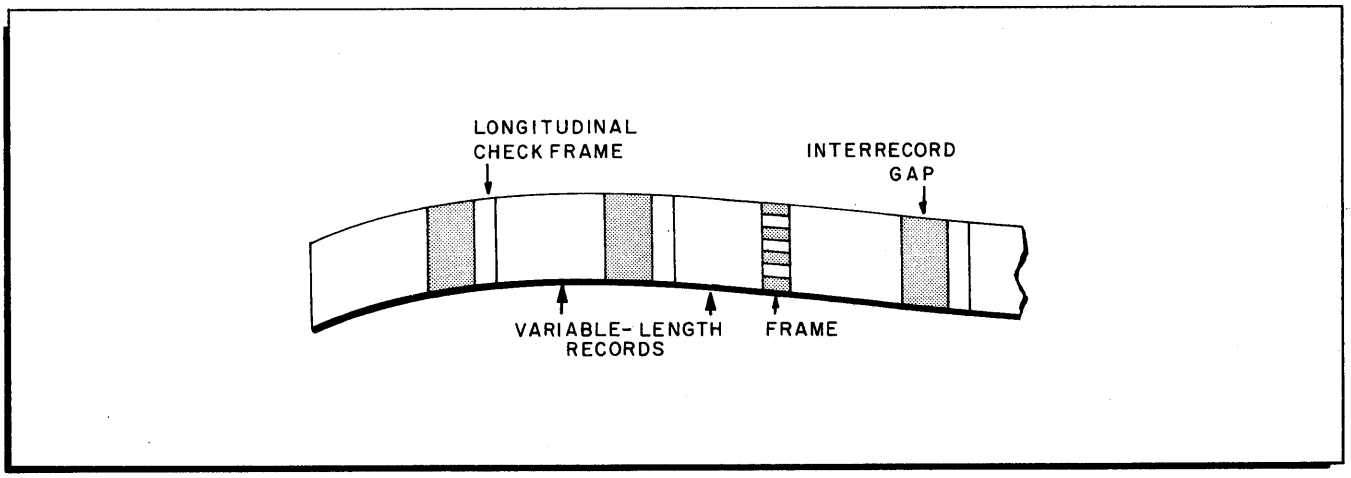


Figure 2-6. Data Format on Magnetic Tape

Punched Card

Punched cards provide a convenient means of entering data into the computer. The cards used for this purpose are either standard 12-row, 80-column cards or 51-column cards. Each card column may contain a decimal digit, an alphabetic character, or a special symbol such as a slash, an asterisk, etc. (see Figure 2-7). The Model 8200 character codes and punched card equivalents are shown in Table 2-1.

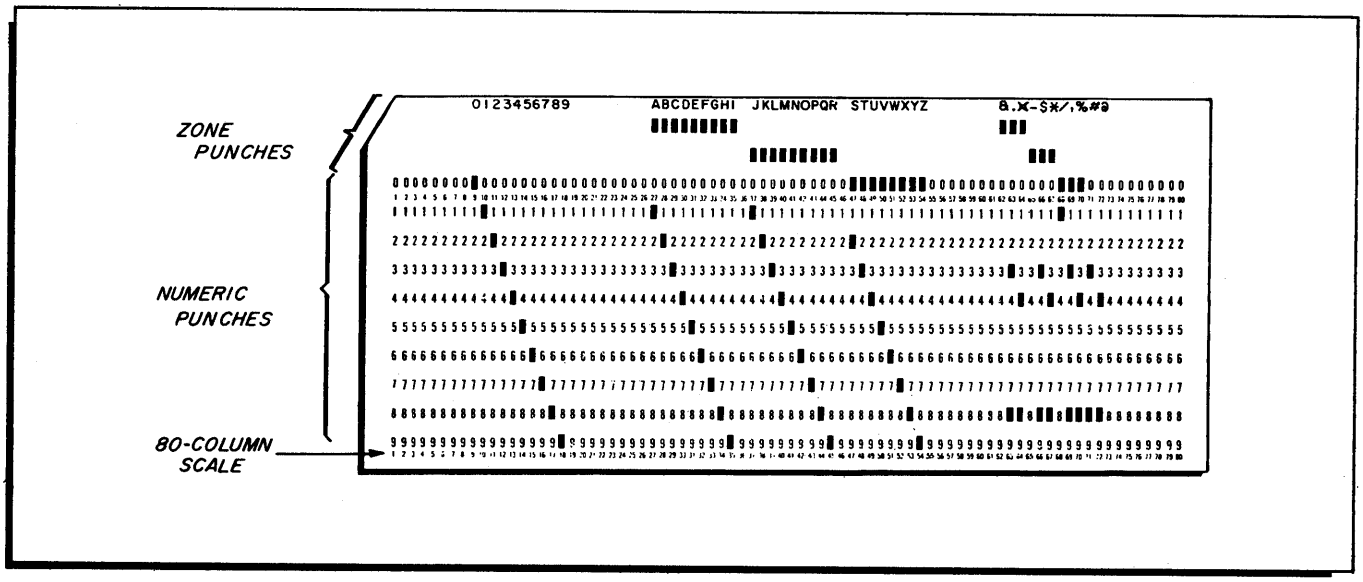


Figure 2-7. Punched Card Format

Numeric information is represented using the card punch positions labeled zero through nine. Alphabetic information is represented by a combination of numeric punches and zone punches. There are three zone punch positions: the 12 zone at the top edge of the card, the 11 zone just below the 12-zone position, and the zero labeled as row zero on the card. The 11 and 12 zones are not labeled because the top edge of the card is reserved for printed headings.

In addition to Hollerith code, cards may be punched or read in direct transcription mode as an optional feature. Each punch position on the card is individually significant in this mode, a punch representing a one bit and the absence of a punch representing a zero bit.

The data formats of the media most commonly associated with peripheral devices (viz., magnetic tape and punched cards) have been described. However, other media (e.g., paper tape, disks, printers, etc.) also contain unique data formats which are converted to or from the processor format by their respective peripheral controls. These formats are described in the individual publications which define such devices.

Table 2-1. Series 200 Character Codes

Key Punch	Card Code	Central Processor Code	Octal	High Speed Printer	Key Punch	Card Code	Central Processor Code	Octal	High Speed Printer
0	0	000000	00	0	0̄ or -	X, 0 or X[1]	100000	40	-
1	1	000001	01	1	J	X, 1	100001	41	J
2	2	000010	02	2	K	X, 2	100010	42	K
3	3	000011	03	3	L	X, 3	100011	43	L
4	4	000100	04	4	M	X, 4	100100	44	M
5	5	000101	05	5	N	X, 5	100101	45	N
6	6	000110	06	6	O	X, 6	100110	46	O
7	7	000111	07	7	P	X, 7	100111	47	P
8	8	001000	10	8	Q	X, 8	101000	50	Q
9	9	001001	11	9	R	X, 9	101001	51	R
	8, 2	001010	12	'		X, 8, 2	101010	52	#
#	8, 3	001011	13	=	$	X, 8, 3	101011	53	$
@	8, 4	001100	14	:	*	X, 8, 4	101100	54	*
Space	Blank	001101	15	Blank	'	X, 8, 5	101101	55	"
	8, 6	001110	16	> [2]		X, 8, 6	101110	56	≠ [2]
&	8, 7	001111	17	&	- or 0̄	X or X, 0[1]	101111	57	1/2 or ![2][3]
0 or &	R, 0 or R[1]	010000	20	+		8, 5	110000	60	< [2]
A	R, 1	010001	21	A	/	0, 1	110001	61	/
B	R, 2	010010	22	B	S	0, 2	110010	62	S
C	R, 3	010011	23	C	T	0, 3	110011	63	T
D	R, 4	010100	24	D	U	0, 4	110100	64	U
E	R, 5	010101	25	E	V	0, 5	110101	65	V
F	R, 6	010110	26	F	W	0, 6	110110	66	W
G	R, 7	010111	27	G	X	0, 7	110111	67	X
H	R, 8	011000	30	H	Y	0, 8	111000	70	Y
I	R, 9	011001	31	I	Z	0, 9	111001	71	Z
	R, 8, 2	011010	32	;		0, 8, 2	111010	72	@
.	R, 8, 3	011011	33	.	,	0, 8, 3	111011	73	,
□	R, 8, 4	011100	34	)	%	0, 8, 4	111100	74	(
	R, 8, 5	011101	35	%		0, 8, 5	111101	75	C_R
	R, 8, 6	011110	36	■		0, 8, 6	111110	76	□ [2]
& or 0 &	R or R, 0[1]	011111	37	?[2]		0, 8, 7	111111	77	¢ [2]

[1] Special Code (for use with H-400/1400 and H-800/1800 cards). The second (alternative) card code is equivalent to the stated central processor code when control character 26 is coded in a card read or punch PCB instruction.

[2] Indicates symbol which will be printed by a printer which has a 63-character drum (Type 222 printers).

[3] The exclamation point replaces the one-half symbol on a type roll containing the Mark II character font.

SECTION III
MEMORY SUBSYSTEM

The memory subsystem is composed of a main memory and a memory controller (see Figure 3-1). Data storage capacities of the memory subsystem range from 262,144 to 1,048,576 characters (32,768 to 131,072 words). Table 3-1 shows the memory configurations that are available with the Model 8200.

Table 3-1. Model 8200 Memory Configurations

MODEL	TOTAL CHARACTERS	TOTAL WORDS	INTERLEAVED ADDRESSING	NUMBER OF MODULES
8201-1	262,144	32,768	No	2
8201-2	524,288	65,536	2-Way	4
8201-3	786,432	98,304	2-Way	4
8201-4	1,048,576	131,072	4-Way	8

MAIN MEMORY

The main memory is a core storage unit that consists of either two, four, or eight independent memory modules. Each module is four characters in width and generally 32,768 four-character groups in length. Cycle time, when memory access is by the word processor, is 750 nanoseconds per 48-bit word; when memory access is by the VLF processor or the input/output subsystem, cycle time is 750 nanoseconds per four characters. Figure 3-2 illustrates the format for both words and characters in main memory.

A parity bit accompanies each character (eight parity bits per word) and parity is checked whenever information is read out of memory. If parity is not odd, an error occurs and control is transferred to the master control facility.

As shown in Figure 3-3, memory can be addressed on either a word or character basis. When memory is addressed on a character basis, a 4-character group containing the addressed character is delivered to either the VLF processor or the input/output subsystem. The delivery of four characters serves to significantly reduce the number of memory references for certain operations and greatly increase the effective operating speed of the system.

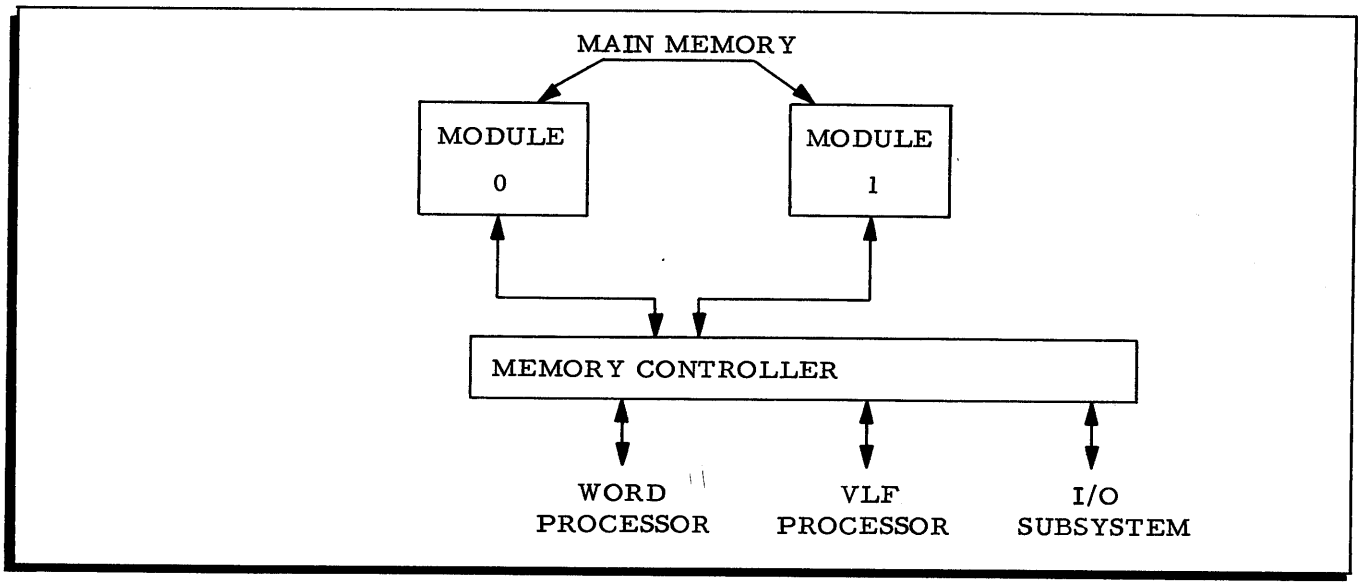


Figure 3-1. Basic Memory Subsystem

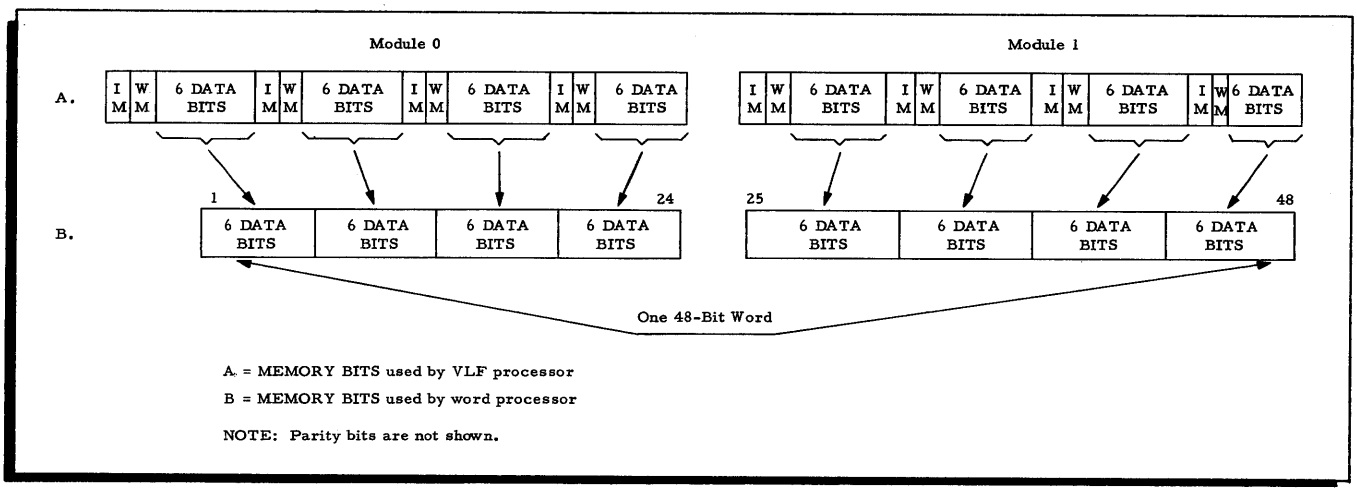


Figure 3-2. Word/Character Memory Format

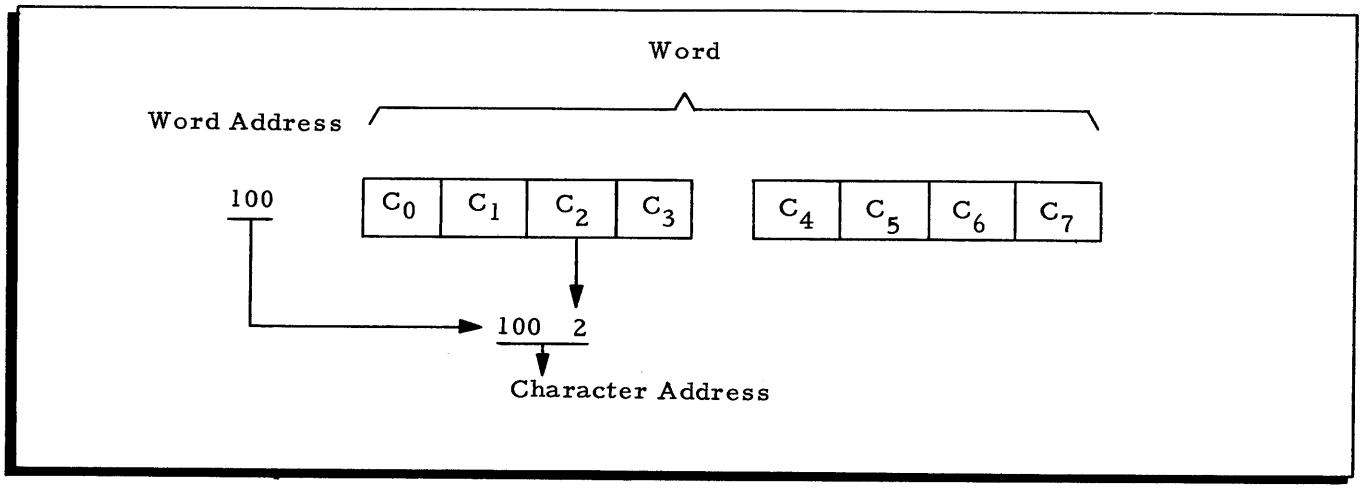


Figure 3-3. Word and Character Addressing

The word processor requires eight characters per memory reference to form each 48-bit word desired (see Figure 3-2). This is accomplished by automatically addressing a pair of memory modules to form one complete word. The memory subsystem is constructed so that both references proceed in parallel and the entire word is delivered to the word processor in the time required for one memory cycle. When the word processor addresses memory, it receives six data bits and one parity bit per character. The punctuation bits, needed for the operation of the VLF processor and the input/output subsystem, do not affect the operation of the word processor. Data bits are common to both processors; therefore, the word processor and the VLF processor may communicate via memory if desired.

MEMORY CONTROLLER

The memory controller provides for maximum simultaneity of memory operations by its ability to transfer information to or from any four memory modules simultaneously. This ability enables word processing, VLF processing, and input/output operations to proceed independently and concurrently. If either processor or the input/output subsystem requests access to the same module simultaneously, the memory controller always grants priority first to the input/output subsystem, and then, alternately, to either the word or VLF processor. This alternating arrangement serves to avoid the possibility of one processor dominating the use of a single memory module. It should be noted that the word processor never has a memory access conflict with itself, since each memory module will contain either all upper, halves or all lower halves of stored words.

INTERLEAVED ADDRESSING

In order to achieve the simultaneity described above, an interleaved addressing scheme has been incorporated in the hardware of the memory subsystem. This method of addressing is accomplished by assigning successive addresses to different modules so that a program written in a normal sequential manner will address different modules as it proceeds. For example, in the fully expanded memory there are four pairs of memory modules which enable the word processor to achieve four-way interleaving of accesses. With four pairs of modules, word addresses 0, 4, 10,....., are assigned to the first pair of modules; 1, 5, 11,....., are assigned to the second pair of modules, etc. (see Figure 3-4). The use of the interleaved memory permits faster program execution by increasing the probability of the word processor, VLF processor, and I/O controller simultaneously requesting access to separate modules of memory.

In addition, system performance is further increased by allowing a processor to overlap many of its memory operations. This is accomplished by a unique design of the addressing circuitry of the Model 8200. Although 750 nanoseconds are required to cycle main memory, the

addressing and data path circuitry of a processor is only used for a portion of the cycle time (approximately 250 nanoseconds). Therefore, the processor's addressing circuitry is available for another memory access before the first access is completed. By interleaving, the instructions and operands will have been distributed among the available modules; therefore, the processor is able to overlap the instruction fetch and decode with computing, and to overlap the A and B operand fetch. As an example, Figure 3-5 shows the amount of memory overlap possible with the word processor.

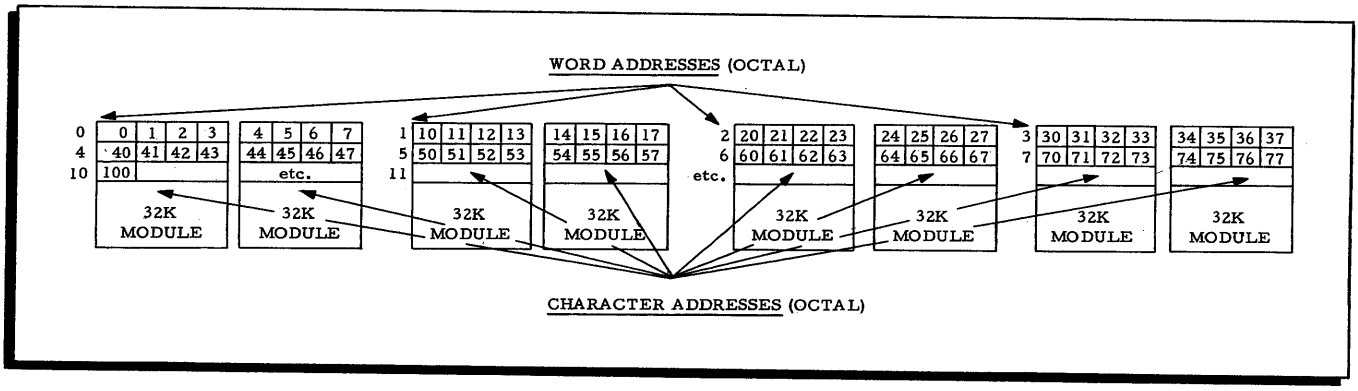


Figure 3-4. Interleaved Addressing Scheme on Fully Expanded System

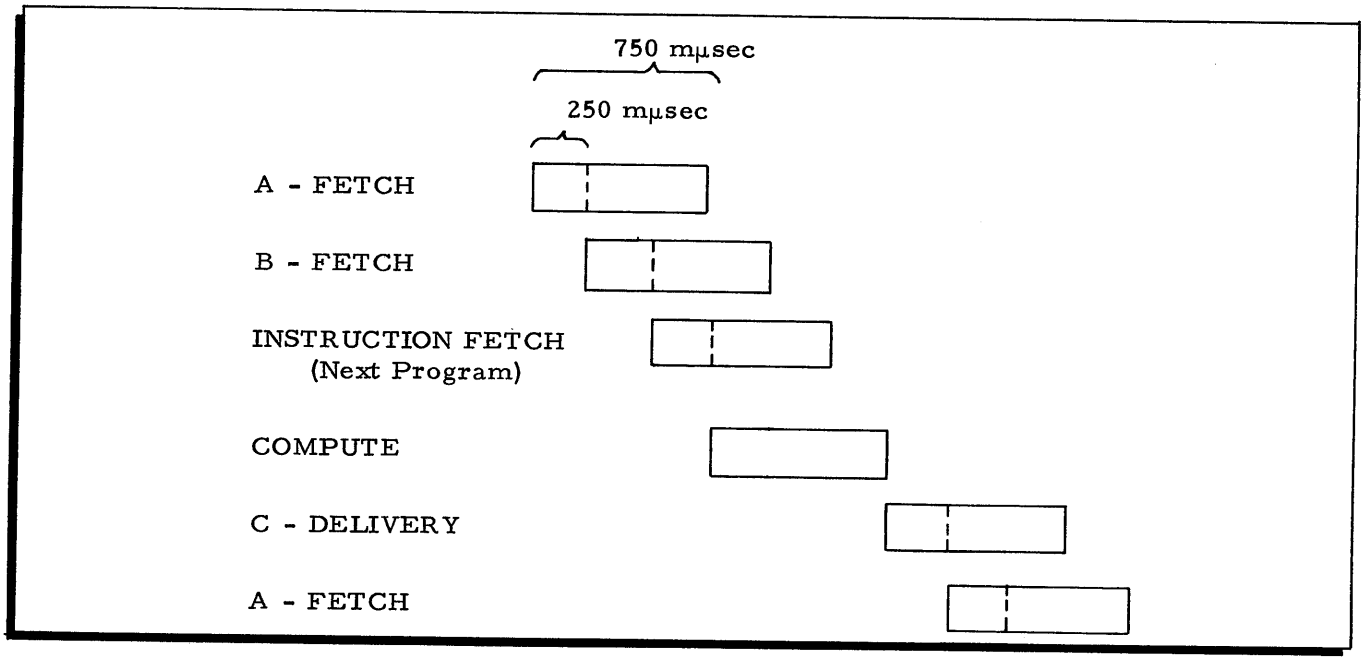


Figure 3-5. Word Processor Memory Overlap

MEMORY PROTECTION

The memory subsystem allows protected memory areas to be designated in blocks of 4,096 characters (512 words). Memory protection is implemented by a system of "locks" and "keys." Each block of memory has a 4-bit lock register and each active element of the Model 8200 has a

4-bit key register, all of which are under exclusive control of the master control facility (see Figure 3-6). Locks may be set to allow either write, read/write, or no protection for each block of memory. Every memory access, whether to extract an instruction, fetch an operand, deliver a result, or transfer via the input/output subsystem, requires the use of a key. The memory subsystem compares the key against the lock of the addressed memory location and if a match exists, the memory access is allowed. If the key does not match the lock, the memory controller does not allow the access but instead reports the attempted violation to the master control facility for resolution.

Since each active element in the Model 8200 has its own key register, the VLF processor and each of the eight groups of the word processor can be given their own private areas of main memory.[1] In this way, the Model 8200 can be divided into as many as nine separate and independent computers. If each of the locks is set to allow read/write protection, programs cannot interfere with each other. However, the master control facility, which coordinates the operation of both processors and the input/output subsystem, has access to any area of memory at any time.

In addition to the above, incorporated in the memory subsystem is a second type of memory protection for the VLF processor. This protection allows the portion of memory allocated to the VLF processor to include a relocatable, two-fence barricade system. This system partitions memory so that an active VLF program is prevented from unintentionally interfering with other VLF programs resident in memory. A more detailed discussion of this protection feature can be found in Section VI.

BASE RELOCATION REGISTER

Each of the eight program control groups in the word processor and the VLF processor is equipped with a base relocation register. The base relocation register functions as a basic index register to provide ease in program relocation. All addressing is done relative to the contents of the base relocation register, and only the master control facility has control over its contents. The contents of the base relocation register are added to the address called for by the program for each memory access. A program running in either processor need not be aware of its actual location in memory nor aware of any other programs that may be resident in memory. The actua location of a program(s) in memory is under control of the master control facility.

[1] It should be noted that there is no restriction on the number of blocks of main memory that may be assigned to each active element except, of course, the total memory size.

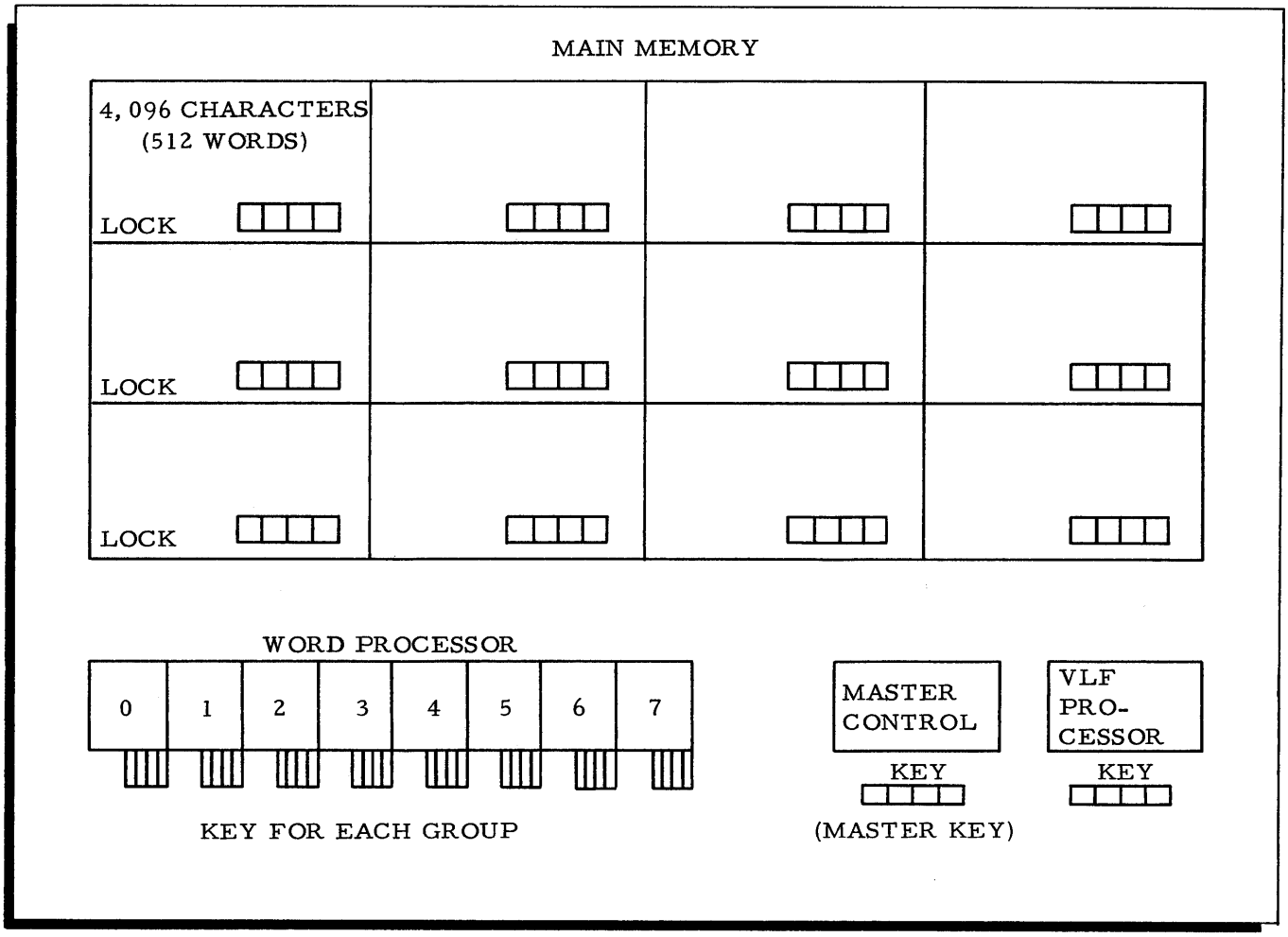


Figure 3-6. Memory Protection

Of great importance is the fact that the master control facility can relocate a running program within memory whenever convenient. This allows a significant improvement in the utilization of the system since the master control facility can function to attain the best possible use of main memory.

STOPPER RELOCATION REGISTER

Each of the eight control groups in the word processor is provided with a stopper relocation register. This register is set by the master control facility when the control group's main memory assignment is made. The contents of the stopper relocation register, called the stopper address, normally represents the highest-numbered location in main memory allocated to a particular program.

The stopper address by definition is neither incremented or decremented when it appears in a program control register. By addressing the stopper in a read instruction, it is possible to

move tape without disturbing any memory locations except the stopper. Similarly, it is possible to read only part of a record into memory and discard the balance by causing the first unwanted word to be stored in the stopper location.

Although the stopper address cannot be incremented, it is possible for an address in a control register to receive an increment or augment greater than the difference between its initial value and the stopper or to be decremented or negatively augmented by an amount greater than its initial value. Since the stopper address is normally the upper limit of a program's memory allocation, any of the above conditions may constitute a memory protection violation and result in a call to master control.

SECTION IV
INPUT/OUTPUT SUBSYSTEM

As is the case with main memory, the input/output subsystem is shared between the word processor and the VLF processor. This subsystem consists of an input/output controller and a wide variety of peripheral devices.

INPUT/OUTPUT CONTROLLER

The input/output controller directs all peripheral device activity. It provides the path through which peripheral commands are initiated and the means for controlling the resulting data transfers. The controller consists of either 32 or 48 variable-speed read/write channels which provide facilities for up to 34 simultaneous input/output operations between the processors and up to 96 peripheral controls. A pair of control registers is provided for each read/write channel which allows the controller to keep track of the starting and current memory locations used by each active I/O operation. Thus, I/O operations may proceed with complete independence once they are initiated.

The input/output controller allows each peripheral device to function with either processor. When the VLF processor reads or writes on a device, the controller functions on a character basis. When the word processor reads or writes on a device, the controller functions on a word basis by counting off groups of eight characters each to form one complete word. Data transfers between peripheral devices and main memory via the controller are simultaneous with processor operations. Memory accesses by the controller do not delay the processors unless a memory conflict occurs (i.e., simultaneous demand for the same memory module), in which case the memory controller grants priority to the I/O controller.

The basic input/output controller consists of two sets of 16 variable-speed read/write channels; up to 16 channels in any combination can be used simultaneously. One set performs the read/write functions for the word processor while the other set performs the same functions for the VLF processor. When Feature 8214 is included, the input/output controller provides the word processor with 16 additional channels for a total of 48. With this expanded system, up to 34 channels can be used simultaneously for data transfer operations.

Sectors

The basic input/output controller is divided into three sectors. Each sector has facilities which allow permanent connection of up to 16 peripheral controls. With Feature 8214, the

number of sectors is increased to six, thus allowing the permanent connection of up to 96 peripheral controls (16 per sector). In order to provide the maximum number of simultaneous input/output operations, the I/O controller does not permanently associate particular read/write channels with particular sectors. Instead, these channels are allowed to "float" from sector to sector so as to assure their maximum utilization by the system.

The basic data transfer rate of the input/output controller is 1, 333, 333 characters per second. This data transfer rate is shared by the three sectors in following manner: sectors 1 and 2 each have a maximum data transfer rate of 500, 000 characters per second, while sector 3 has a maximum rate of 333, 333 characters per second. Sectors 1 and 2 each provide facilities to permit up to six peripheral controls to simultaneously share its 500KC data transfer rate. Thus, up to six devices per sector can operate concurrently provided each device does not exceed a data transfer rate of 83, 333 characters per second. Sector 3 simultaneously handles up to four peripheral devices and also has a maximum data transfer rate of 83, 333 characters per second per device when the four devices are operated concurrently.

When Feature 8214 is included, sectors 1 and 3 remain unchanged but sector 2 is replaced with four buffered sectors (numbered 2A, 2B, 2C, and 2D). Each buffered sector has a data transfer capacity of 500, 000 characters per second, can handle up to six peripheral devices simultaneously, and provides facilities to permanently attach up to 16 peripheral controls. In addition, when Feature 8214 is included, the input/output controller can accommodate a peak data transfer rate of 2, 833, 333 characters per second.

A single buffered sector is equipped with six 4-character buffers. Therefore, up to six devices operating concurrently are provided with a 4-character storage area. A buffer accumulates up to 4 characters of data before requiring access to main memory. The buffered sectors may be used without their buffer areas (direct mode) but this arrangement results in a slower data transfer rate. Table 4-1 indicates whether or not a control/device can be connected to a buffered sector in either the buffered or the direct mode.

In order to attain optimum system performance, sectors and their maximum data transfer rates should be taken into consideration before permanently connecting peripheral controls to paritcular sectors.

Read/Write Channels

The input/output data path between the I/O controller and a peripheral control is completed by one or more "read/write channels," which are inserted in the data path when the input/output instruction is executed.

Table 4-1. Controls/Devices Connectable to Buffered Sectors

CONTROL/DEVICE	BUFFERED MODE	DIRECT MODE
Type 207, 208 Control with Type 227 Card Reader/Punch	No	No
Type 208-1, -2 Control with Type 224-1, -2 Card Reader/Punch	Yes	Yes
Type 223 Card Reader and Control	Yes	No
Type 214-1 Card Punch and Control	Yes	No
Type 214-2 Card Reader/Punch and Control	Yes	No
Type 209, 209-2 Paper Tape Reader and Control	No	Yes
Type 210 Paper Tape Punch and Control	Yes	Yes
Type 206 Printer and Control	No	No
Type 222-1, -2, -3, -4, -5 Printer and Control	Yes	No
Type 203 Tape Controls (all)	Yes	No
Type 281 Single-Channel Communication Controls (all)	No	Yes
Type 286 Multi-Channel Communication Controls (all)	No	No
Type 287 Autodin Communication Control	No	Yes
Type 233-2 MICR Control	No	Yes
Type 234 Plotter Control	Yes	Yes
Type 212 On-Line Adapter	No	No
Type 257 Control for Type 258 and 259 Disk Drives	Yes	No
Type 260 Control for Type 261 and 262 Disk Files	Yes	No
Type 8201-3 Console[1]	No	No

[1]The 8201-3 Console must be connected to sector 1.

Read/write channels (RWC's) in the input/output controller are variable-speed channels which permit the use of a wide array of peripheral devices with varying data transfer rates. Normally during a data transfer operation, one character of information is transferred by a read/write channel to or from main memory during one memory cycle. However, to achieve the higher data transfer rates of certain peripheral devices, the input/output controller traffic control offers variable numbers of memory cycles per unit of time to each channel, depending upon the read/write channel assignment code used in the instruction which initiates the operation (see

Table 4-2). From one to six cycles are offered to a read/write channel every 12 microseconds, giving channel data transfer capacities ranging from 83,333 to 500,000 characters per second. Effectively, then, the input/output controller incorporates variable-speed read/write channels.

Table 4-2. Read/Write Channels and Corresponding Data Transfer Rates

SECTOR	DATA TRANSFER RATE (thousands of characters per second)	READ/WRITE CHANNEL[1]
1	83	1
	83	2
	83	3
	83	1A
	83	2A
	83	3A
	250	2
	167	1 1A
	167	2 2A
	167	3 3A
	500	3
	250	3
	333	3
2	83	4
	83	5
	83	6
	83	4A
	83	5A
	83	6A
	250	5
	167	4 4A
	167	5 5A
	167	6 6A
	500	6
	250	6
	333	6
3	83	8
	83	9
	83	8A
	83	9A
	167	8 8A
	167	9 9A
	333	9

[1] Underlined RWC is used for address storage during data transfer. Non-underlined RWC's are busy if interlocked with a busy underlined RWC.

No more than two RWC's (a primary and the corresponding auxiliary) are ever made busy by a single RWC assignment. However, a single RWC assignment can still command a data transfer capacity of up to 500,000 characters per second. The most important advantage of this

arrangement is that the RWC's not made busy by a high-speed transfer are available for use in other operations. For example, in order to handle a 250,000-character-per-second input/output transfer, only one primary channel will be tied up while the other RWC's in the same sector will still be available for use in other operations, e.g., three 83,333-character-per-second transfers. However, in no case can the total data transfer rate of a single sector exceed 500,000 characters per second (333,333 characters per second for sector 3).

As previously mentioned, RWC's are allowed to float from sector to sector. This capability permits a RWC normally restricted to operating in one sector to be used for I/O operations in another sector. For example, RWC 1, normally used in sector 1, can be assigned to sector 2 input/output operations. It should be noted that this capability does not increase a particular sectors maximum data transfer rate.

Read/Write Channel Assignment

The primary consideration in selecting a RWC assignment for a particular application is the rated speed at which the device being addressed transfers data to or from main memory. The one or two RWC's assigned to an operation must receive memory accesses often enough to keep up with the I/O data transfer rate of the device. For example, the RWC assignment used in a data transfer instruction which addresses a Type 258 Disk Pack Drive must designate a data transfer capacity high enough to keep pace with the device's nominal 208,000-character-per-second transfer rate.

However, due to mechanical tolerances, some devices may transfer data at instantaneous rates higher than their nominal transfer rates. In such cases, the devices require an RWC assignment having a greater data handling capacity than would be required if the nominal data transfer rate were maintained. As an example, a Type 204B-5 tape drive using a density of 556 bits per inch requires an RWC assignment having a data handling capacity of 167,000-characters per second, even though the nominal transfer rate for this device is less than 83,000 characters per second.

Table 4-3 lists the minimum RWC capacity requirements for each Series 200 peripheral device.

Input/Output Protection

Memory protection is extended to peripheral operations which makes it possible to ensure that peripheral transfers are subject to lock and key checking. A peripheral protection table is assembled for each active program by the master control facility. When an I/O operation is initiated by a program, the control and device addresses are automatically checked against the

table in order to verify that the program has been allocated the addressed units. If the addresses are not found in the table, the I/O operation is not performed and master control is called to resolve the illegal request. In addition, the table is used to verify that the data transfer operation is directed to the main memory area assigned to the program. The table is updated periodically by master control to allow devices and controls released by a completed program to be assigned to other active programs.

Table 4-3. Minimum RWC Capacity Requirements for Series 200 Peripheral Devices

DEVICE	MINIMUM RWC CAPACITY REQUIRED (characters per second)	DEVICE	MINIMUM RWC CAPACITY REQUIRED (characters per second)
204A-1 Magnetic Tape Unit	83.3 KC	209 Paper Tape Reader	83.3 KC
204A-2 Magnetic Tape Unit	167 KC	209-2 Paper Tape Reader	83.3 KC
204A-3 Magnetic Tape Unit	167 KC	210 Paper Tape Punch	83.3 KC
204B-1, -2 Magnetic Tape Units		212 On-Line Adapter	167 KC
200/556 bpi	83.3 KC		
204B-3, -4 Magnetic Tape Units		212-1 Central Processor Adapter	167 KC
200/556 bpi	83.3 KC		
204B-3, -4 Magnetic Tape Units		213-4 Time-of-Day Clock	83.3 KC
200/556 bpi	83.3 KC		
204B-5 Magnetic Tape Unit		220-1, -2, -3 Consoles	83.3 KC
200 bpi	83.3 KC		
556 bpi	167 KC		
204B-7 Magnetic Tape Unit		234 Calcomp Plotter Control	83.3 KC
200/556/800 bpi	83.3 KC		
204B-8 Magnetic Tape Unit		235 Optical Journal Reader Control	83.3 KC
200/556 bpi	83.3 KC		
800 bpi	167 KC		
204B-9 Magnetic Tape Unit		237 Bill Feed Printer Control	167 KC
200/556 bpi	83.3 KC		
800 bpi	167 KC		
204B-11, -12 Magnetic Tape Units		258 Disk Drive	250 KC
200/556 bpi	83.3 KC		
204C-13, -14 Magnetic Tape Units	83.3 KC	259 Disk Drive	250 KC
206 Printer	167 KC	259A Disk Drive	167 KC
214-1 Card Punch	83.3 KC	261 Disk File	250 KC
214-2 Card Reader/Punch		262 Disk File	250 KC
Read	83.3 KC		
Punch	83.3 KC		
222 Printers (All Models)	167 KC	270 Random Access Drum Storage (All Models)	167 KC
223 Card Reader	83.3 KC	281 Single-Channel Communication Controls[1] (All Models)	83.3 KC
224-1, -2 Card Reader/Punch		286-1, -2, -3 Multi-Channel Communication Controls	83.3 KC
Read	83.3 KC		
Punch	83.3 KC		
227 Card Reader/Card Punch		286-4, -5 Message-Mode, Multi-Channel Communication Controls[2]	83.3 KC
Read	167 KC		
Punch	167 KC		
233-2 MICR Control	83.3 KC	287 Autodin Communication Control[1]	83.3 KC

[1] Both the 281-2F and 287 controls require exclusive assignment of two 83.3 KC RWC's when operating in full-duplex mode.

[2] The maximum RWC capacity that can be assigned to a 286-4 or 286-5 is 167 KC.

Also, master control has the facilities to alter a program's request for a particular device. As an example, if a program specifies a certain tape unit as a work tape and the tape unit is not available (i.e., in use, off line, etc.), then master control may assign another similar tape unit or even a disk device to the program. Additionally, to increase their utilization, peripheral controls and devices may be shared by active programs in both processors.

Steering Registers

Another feature of the I/O controller allows the VLF processor to bypass master control intervention and function directly with a peripheral control. This is accomplished by providing the controller with up to 96 steering registers (one per control) which can be set or reset only by master control. Once a steering register is set, the corresponding control may issue interrupts directly to or receive peripheral commands directly from the VLF processor. This type of arrangement is useful in communication and real-time applications.

In addition, the steering registers also indicate whether a control can be shared between processors. When a steering register is set, only the VLF processor can use the corresponding peripheral control.

Maintenance Processor and Peripheral Control Switch

In addition to the normal functions of input/output data transferring, the I/O controller also provides facilities for independent maintenance operations on peripheral controls and devices. This feature is implemented by a maintenance processor and peripheral control switch. With these facilities, a service engineer may switch off-line any control connected to the Model 8200 and perform maintenance on the control itself or any device connected to the control. The maintenance processor contains logic which simulates the peripheral commands normally issued by the processors. This maintenance can be performed without affecting the operation of the remaining system components. Availability of this type of maintenance will reduce the effect of peripheral malfunctions on total system performance.

PERIPHERAL EQUIPMENT

The array of peripheral devices available with the Model 8200 includes over 40 units: punched card equipment, high-speed printers, magnetic tape units, paper tape equipment, random access drum units, disk pack drives, disk files, and various data communication equipment. Also included are Visual Information Projection units, teller terminals, Audio Response Systems, computer-to-computer adapters, an interval timer, a time-of-day clock, an MICR sorter-reader control, an optical journal reader control, a digital plotter control, a bill feed printer control, magnetic tape switching units, and communication control switching units which provide extremely flexible Model 8200 configurations.

Information is transferred between any one of these devices and the input/output controller by means of a single stored-program instruction — the Peripheral Data Transfer instruction. By coding various control characters in this instruction, the programmer specifies the direction of data transfer, the specific device involved in the transfer, the data path over which information is to be transferred (RWC), and any other information necessary to define the input/output operation (e.g., the number of lines to be spaced during printer operations). The actual communication with the input/output controller is not made by the particular device but by the peripheral control connected to that device.

Peripheral Control

A peripheral control regulates the transfer of data between the input/output controller and a peripheral device. The control compensates for the difference in the data transfer rates of the controller and the device by temporarily storing each character of transmitted information until either the controller or the device is ready to receive the character. The control also converts each character into the code used by the intended recipient (e.g., the card reader control converts a character from Hollerith code to the internal 6-bit code of the processors). As each character is transferred to the control, it is also checked for accuracy by the control. One particularly significant feature of the control is that it operates independently and requires access to main memory only when information transfers are performed. In particular, all of the previously mentioned activities of the control — temporarily storing, converting, and checking the information — do not involve the input/output controller in any way. When each character of information is transferred, one main memory cycle is allocated for the transfer.

Some peripheral devices require one peripheral control per device (e.g., a card reader). Other devices can be connected in multiple fashion to a single control (e.g., up to eight tape units can be directed by a single control). The number of devices connectable to a peripheral control is shown in Tables 4-4 through 4-14. The information listed under "Unit Loads" and "Address Assignments" in this table is used in determining the number of peripheral controls that can be connected to the Model 8200, as explained below.

Peripheral Addresses and Unit Loads

When installed in a Model 8200 system, peripheral controls (and their associated devices) are permanently connected to the system. Each control is assigned one or two addresses, depending on the number of directions in which it can transfer data. It is by these peripheral addresses that the controls are designated in input/output instructions. For example, a card reader and its associated control can transfer data in only one direction — into the input/output controller. The reader is therefore assigned one address by which it is always designated in an instruction. A combination card reader/punch and control can transfer data into two directions —

in and out of the controller. It is thereby assigned two addresses: one address is used to specify an input (card read) operation, while the other is used to specify an output (card punch) operation.

The number of peripheral controls which the Model 8200 can accommodate depends upon four factors: (1) the number of "unit loads" of power required by the controls to be connected; (2) the number of unit loads available from the processor; (3) the number of peripheral addresses required to operate the controls; and (4) the number of address assignments which the processor provides. Up to 96 unit loads and 96 address assignments are available with the Model 8200. The number of unit loads and address assignments required by each control is shown in Tables 4-4 through 4-14.

Punched Card Equipment

The Model 8200 utilizes a wide variety of peripheral devices not only of different kinds but also on several performance levels for the same kind. For instance, six different punched card units are offered: a card reader, a card punch, and four reader/punches. Table 4-4 lists the card devices available with the Model 8200. Note that a card device requires either one or two "unit loads," depending on the number of functions the device performs.

Table 4-4. Punched Card Equipment

Device		Read/Punch Speed	No. Devices Per Control	No. Unit Loads Required by Control & Device	Address Assignments
Type	Function				
223	Card Reader	800 cards/minute	1	1	1
214-1	Card Punch	100-400 cards/minute	1	1	1
224-1	Card Reader/Punch[1]	Read: 300 cards/minute Punch: 50-270 cards/minute	1	1	2
224-2	Card Reader/Punch[1]	Read: 400 cards/minute Punch: 90-360 cards/minute	1	1	2
214-2	Card Reader/Punch	Read: 400 cards/minute Punch: 100-400 cards/minute	1	1	2
227	Card Reader/Punch	Read: 800 cards/minute Punch: 250 cards/minute	1	2	2

[1] Requires one address assignment when connected for use as a card punch only.

High-Speed Printers

Five types of printers (see Table 4-5) produce printed reports, listings, etc., at speeds which vary from 450 to 1,300 lines per minute. Processed information is printed from any programmer-assigned area in memory.

Table 4-5. High-Speed Printers

Type	Print Speed	No. Printers Per Control	No. Unit Loads Required by Control & Device	Address Assignments
222-1 (96 print positions)	650-1,300 lines/minute	1	1	1
222-2 (108 print positions)	650-1,300 lines/minute	1	1	1
222-3 (120 or 132 print positions)	650-1,300 lines/minute	1	1	1
222-4 (120 or 132 print positions)	950-1,266 lines/minute	1	1	1
222-5 (120 print positions)	450 lines/minute	1	1	1

Magnetic Tape Units

Magnetic tape is a compact and highly versatile medium for the storage of programs and data files. Two complete families of industry-acclaimed tape units are available with the Model 8200 (see Table 4-6): 1/2-inch tape units (10 types) transfer data at speeds ranging from 4,800 to 96,000 characters per second; three types of 3/4-inch tape units read/write from 32,000 to 88,800 characters per second. The capability of processing nine-track, 1/2-inch tape is also provided.

Table 4-6. Magnetic Tape Units

Type	Data Transfer Rate	No. Devices Per Control	No. Unit Loads Required by Control & Devices	Address Assignments
1/2-Inch Magnetic Tape Units				
204B-1 204B-2	7,200/20,000 characters/second	1-8	2	2
204B-3 204B-4	16,000/44,400 characters/second	1-8	2	2
204B-5	24,000/66,700 characters/second	1-8	2	2
204B-7	20,000/28,800 (or 7,200/28,800) characters/second	1-8	2	2
204B-8	44,400/64,000 (or 16,000/64,000) characters/second	1-8	2	2
204B-9	66,700/96,000 (or 24,000/96,000) characters/second	1-8	2	2

Table 4-6 (cont). Magnetic Tape Units

Type	Data Transfer Rate	No. Devices Per Control	No. Unit Loads Required by Control & Devices	Address Assignments
1/2-Inch Magnetic Tape Units				
204B-11 204B-12	4,800/13,300 characters/second	1-4	2	2
204C-13 204C-14	28,800 characters/second	1-2	2	2
3/4-Inch Magnetic Tape Units				
204A-1	32,000 characters/second	1-4	2	2
204A-2	64,000 characters/second	1-4	2	2
204A-3	88,800 characters/second	1-4	2	2

Disk Pack Drives

Honeywell Disk Pack Drives combine the desirable features of magnetic tape and magnetic disk storage — unlimited shelf storage and fast random access. This is made possible by the use of removable disk packs which may be recorded on, stored indefinitely (like magnetic tape), and rapidly reinserted in an on-line drive. The various Disk Pack Drives are listed in Table 4-7.

Table 4-7. Disk Pack Drives

Type	Data Storage Capacity Per Drive	Data Transfer Rate	No. Devices Per Control	No. Unit Loads Required by Control & Devices	Address Assignments
258	4.6 million characters	208,333 characters/second	1-8	1	2
259	9.2 million characters	208,333 characters/second	1-8	1	2
259A	9.2 million characters	147,500 characters/second	1-8	1	2

Disk Files

The Honeywell Disk Files are fixed-disk storage devices which provide an extremely high on-line storage capacity (see Table 4-8). A single Disk File subsystem's capacity may amount to over 1.2 billion characters. Any on-line data track can be located in a maximum time of 120 milliseconds, and data can be transferred at a rate of 190,000 characters per second.

Table 4-8. Disk Files

Type	Data Storage Capacity Per File	Data Transfer Rate	No. Devices Per Control	No. Unit Loads Required by Control & Devices	Address Assignments
261	150 million characters (nominal)	190,000 characters/second	1-8	1	2
262	300 million characters (nominal)	190,000 characters/second	1-4	1	2

Random Access Drums

The Model 8200 drum storage capability features a drum control which can direct from one to eight magnetic drums, each capable of storing 2.6 million characters of information (see Table 4-9). Thus, a single drum subsystem can have a total capacity of over 20 million characters. Any record stored on the drum can be located in 31 milliseconds (average) and can be transferred at the rate of 111,000 characters per second.

Table 4-9. Magnetic Drum Units

Type	Data Storage Capacity Per Drum	Data Transfer Rate	No. Devices Per Control	No. Unit Loads Required by Control & Devices	Address Assignments
270A-1 through 270A-8	2.6 million characters	111,000 characters/second	1-8	1	2

Paper Tape Equipment

Paper tape is an ideal medium for recording data which originates at locations distant from a central installation and, as such, becomes particularly significant in data communication networks. A variety of standard commercial codes may be used with this relatively inexpensive medium. Two paper tape devices are offered with the Model 8200 (see Table 4-10).

Data Communication Equipment

Communication controls allow the Model 8200 to communicate with distant locations (e.g., branch offices, warehouses, etc.) by receiving and transmitting data over toll and leased lines. Three kinds of communication controls are available: (1) four types of single-channel controls transfer entire messages over single lines; (2) three types of multi-channel controls transfer messages character-by-character over as many as 63 different lines; and (3) two types of

message-mode multi-channel controls transfer entire messages over a maximum of 63 lines. All controls are adaptable to a broad selection of lines, speeds, and terminal devices. One such terminal device is Honeywell's Data Station (see Table 4-11).

Table 4-10. Paper Tape Equipment

Device		Data Transfer Rate	No. Devices Per Control	No. Unit Loads Required by Control & Devices[1]	Address Assignments
Type	Function				
209	Paper Tape Reader	600 characters/second	1	2	1
210	Paper Tape Punch	120 characters/second	1	2	1

[1] The total power requirement for the combination of a 209 reader and a 210 punch is 3 unit loads.

Table 4-11. Data Communication Equipment

Device		Data Transfer Rate	No. Devices Per Control	No. Unit Loads Required by Control & Devices	Address Assignments
Type	Function				
		Communication Controls			
281-1, -2, -3, and -4	Single-Channel Controls	Up to 5,100 characters/second	1 line	1[1]	2
286-1, -2, and -3	Multi-Channel Controls	Up to 300 characters/second/line	1-63 lines	2	2
286-4, -5	Message-Mode Multi-Channel Controls	Up to 7,000 characters/second (all lines)	1-63 lines	2	2
287	AUTODIN Communication Control	Up to at least 4800 baud	1 line	2	2
		Remote Terminal Device			
288-1	Data Station Central Control	120 characters/second	not applicable	not applicable	not applicable
288-3	Data Station Central Control[2]	300 characters/second	not applicable	not applicable	not applicable
289-2	Data Station Page Printer & Keyboard	10 characters/second	not applicable	not applicable	not applicable
289-2A	Keyboard	10 characters/second	not applicable	not applicable	not applicable
289-3	Data Station Page Printer & Keyboard	40 characters/second	not applicable	not applicable	not applicable
289-4	Data Station Paper Tape Reader	120 characters/second	not applicable	not applicable	not applicable

Table 4-11 (cont). Data Communication Equipment

Device		Data Transfer Rate	No. Devices Per Control	No. Unit Loads Required by Control & Devices	Address Assignments
Type	Function				
Remote Terminal Device					
289-5	Data Station Paper Tape Punch	120 characters/second	not applicable	not applicable	not applicable
289-7	Data Station Card Reader	120 characters/second	not applicable	not applicable	not applicable
289-8	Data Station Optical Bar Code Reader	50 characters/second	not applicable	not applicable	not applicable
289-9	Data Station Remote Line Printer	400 lines/minute	not applicable	not applicable	not applicable

[1]The 281-2 control requires two unit loads.

[2]The 289-9 printer requires the 288-3 control and, vice versa, the 289-9 printer must be part of the system if the 288-3 control is used.

Visual Information Projection Devices

A complete CRT display capability — for businesses requiring instantaneous visual access to data stored in computer files — is available to the Model 8200 user. These devices operate on line to the computer, either locally via direct physical connection or from remote locations via communication facilities. As shown in Table 4-12, the CRT devices provide a variety of message characteristics: numeric, number/block alpha, and typewriter. An 8-bit code (7-bit ASCII plus parity) is used for synchronous transmission and a 10-bit code (7-bit ASCII plus parity and start and stop bits) for asynchronous transmission.

Table 4-12. Visual Information Projection Devices

Device		Data Transfer Rate	No. Devices Per Control	No. Unit Loads Required by Control & Devices	Address Assignments
Type	Function				
303	Display Station (Typewriter keyboard)	1200-2400 baud	12 for basic control, plus 12 for each added expansion module	not applicable	not applicable
311	Display Station (15-key block numeric keyboard)	1200-2400 baud			
312	Display Station (43-key numeric/block alpha keyboard)	1200-2400 baud			

Table 4-12 (cont). Visual Information Projection Devices

Device		Data Transfer Rate	No. Devices Per Control	No. Unit Loads Required by Control & Devices	Address Assignments
Type	Function				
304	Display Station (Navcor electronic typewriter keyboard)	1200-2400 baud	12 for basic control, plus 12 for each added expansion module	not applicable	not applicable
317	Display Station (no keyboard)	1200-2400 baud			

TELLER TERMINAL EQUIPMENT

The Teller Terminal Series permits more efficient banking procedure through on-the-counter, on-line processing of all teller-assigned transactions. The Series includes the Type 370 TELLER-REGISTER[1] Window Machine, which is used by the teller for all his bank transactions, and a remote transceiver that transmits transaction information between the Type 370 and the Model 8200. Data is transmitted asynchronously via a modified ASCII-type code permitting combinations of similarly coded peripheral devices to share common networks. This code consists of a start bit, seven data bits, an odd parity bit, and a stop bit. Specifications of the Type 370 are shown in Table 4-13.

Table 4-13. Teller Terminal Equipment

Device		Data Transfer Rate	No. Devices Per Transceiver	No. Unit Loads Required by Transceiver & Devices	Address Assignments
Type	Function				
370	TELLER-REGISTER	120 characters/second	1, 2, 6, or 10	not applicable	not applicable

Additional Peripheral Devices

A number of other peripheral devices are available with the Model 8200. General characteristics of these devices are shown in Table 4-14.

[1] Registered trademark of Bunker-Ramo Corporation.

Table 4-14. Additional Peripheral Devices

Device		Performance	No. Devices Per Control	No. Unit Loads Required	Address Assignments
Type	Function				
212	On-Line Adapter	120,000 characters/second	1	1	1
212-1	Central Processor Adapter	167,000 characters/second	1	2	2
213-3	Interval Timer	Range: 100 microseconds to 200 milliseconds	1	1	1
213-4	Time-of-Day Clock	Range: 00:00:00.0 to 23:59:59.9 (hours, minutes, seconds, and tenths of seconds)	1	1	1
233-2	MICR Sorter-Reader Control for Burroughs B103	Up to 1,560 documents per minute	1	1	1
234	Plotter Control for Calcomp[1] Plotters	Plotting Speed: Up to 300 increments per second (in any of eight directions)	1	1	1
235	Optical Journal Reader Control for NCR Model 420	26 or 52 lines/second	1	1	1
237	Bill Feed Printer Control for Model 1404 Alphanumeric Printer	600 lines/minute; or up to 800 cards/minute	1	2	2

[1] Registered trademark of California Computer Products, Inc.

SECTION V
WORD PROCESSING SUBSYSTEM

The word processing subsystem consists of a control unit, a control register memory, and an arithmetic unit. The word processor performs decimal and binary fixed- and floating-point arithmetic operations, including multiply and divide.

The control unit with its control memory is the nerve center of the word processor. As the site of multiprogram control, it monitors the time sharing of the word processor to achieve maximum efficiency of operation. In addition to its multiprogramming function, the control unit also selects, interprets, and directs the execution of instructions and governs address selection in both control and main memory.

The control memory contains eight identical groups of program control registers, each group containing 32 registers 24 bits in length. The contents of a register are used to select a full word from main memory. Read/write cycle time of the control memory is 125 nanoseconds. Table 5-1 identifies the program control registers within a group. The control registers enable explicit addressing of all main memory locations and provide facilities for indexed, indirect, and indexed indirect addressing. In addition to the above registers included in each group, there are three special-purpose registers which are only accessible by the master control facility. The first is a key register which contains a master-control-facility assigned key to protected memory and assigned peripheral devices. The second is a base relocation register which stores an assigned base address corresponding to the starting location of the group's program in main memory. All addressing is done relative to the contents of this register. The third register is the stopper relocation register. It stores an address within a group's assigned memory allocation. A more detailed description of these registers can be found in the memory subsystem (see Section III).

The word processing subsystem also includes extensive automatic masking facilities which enable the programmer to handle less-than-word-length operands easily and efficiently.

The arithmetic unit is the portion of the word processor in which arithmetic and comparison operations are performed in accordance with logical rules of the command codes. Arithmetic operations are performed on whole 48-bit words. The word processor has provision for both binary and decimal arithmetic, complete logical abilities, and extensive internal checking. Data conversion instructions are provided to convert between binary and decimal formats.

Table 5-1. Word Processor Control Registers

NUMBER	FUNCTION
1	Sequence Register
1	Cosequence Register
1	Sequence History Register
1	Cosequence History Register
2	Arithmetic Control Registers
8	Index Registers
1	Unprogrammed Transfer Register
1	Mask Index Register
12	General Purpose Registers
4	Input/Output Registers

MULTIPROGRAM CONTROL

Multiprogram control directs the time-sharing of the word processor by the active programs. Each of the eight groups of control registers may direct the execution of an independent program. After a program and its associated control register group are loaded, the designated control register group is activated to direct the selection of instructions. This control register group, and the program it directs, is said to be "on." When the program is completed, the directing control register group is inactivated (turned "off"). Control register groups may be turned on and off independently of each other, either from the console or by program control.

When more than one control register group is active, word processor time is shared among the several programs. The rules under which multiprogram control operates are as follows:

1. When an active control register group is selected, the subsequent time allowed the word processor must be used to select the next instruction to be executed in the program controlled by that group; all succeeding processor time must be devoted to the execution of this instruction until completed.
2. If the instruction is one which does not leave unstored information in the arithmetic or control units, its completion causes multiprogram control to scan or "hunt" for the next active control register group in sequence.
3. If the instruction does leave unstored information in the arithmetic or control unit, scanning or hunting is inhibited and the next instruction is selected from the same program. Hunting is not allowed if:
   a. the instruction generates a two-word result, of which only one word is stored in memory as the result of the instruction execution, e.g., multiply;

b. the execution of the instruction results in a sequence change; in the case of a conditional change, hunting is inhibited only if the condition is satisfied;

c. the instruction has an inactive C address;

d. an unprogrammed transfer takes place as the result of executing the instruction;

e. the instruction is capable of specifying that hunting shall not take place, and does so specify.

For example, if three programs are active, then one instruction is performed in turn from program 1, then 2, then 3, then 1, and so on, as long as all instructions selected allow hunting. Such alternation is temporarily held up if an instruction is selected which does not allow hunting, but it is resumed as soon as an instruction is selected which does allow hunting. Thus, the word processor time is shared on a fairly equal basis among all active programs.

The true power of multiprogram control appears when an active program attempts to execute an instruction which cannot be implemented because of the unavailability of a system component. Suppose, for example, that a Peripheral Control and Transfer (PCT) instruction calling for a tape drive is selected from an active program. The word processor, in attempting to execute this instruction, finds that either that tape drive or its associated control is involved in executing another instruction from the same or some other program. Seeing that the instruction cannot be executed immediately and recognizing the reason therefore, multiprogram control places the control register group in a "stall" or temporary off condition. This condition indicates to multiprogram control that this program, although still active, shall not be allowed any processor time as long as the "stall" indication remains. Master control removes the "stall" (temporary off) condition when the PCT instruction can be accepted by the peripheral control.

The result of the operation of multiprogram control is that there is never idle time in the word processor as long as there is any active program in which an instruction can be executed.

MASKING

In many instances, it is desirable to compare portions of words, to perform arithmetic operations on fields of less than word length, or to select data from within a word for transfer to other memory locations and subsequent processing. All of these operations can be accomplished in the word processing subsystem by versatile masking facilities. A mask may be thought of as a filter word through which other words moving between memory and the arithmetic unit may be made to pass. Like all words expressed in machine language, the

word consists only of binary zeros and ones. However, when used in a mask, these binary numbers take on the following significance. Where the mask contains a one, the corresponding bit of the word being moved is allowed to pass; where it contains a zero, the corresponding bit is not allowed to pass.

The word processor provides two types of masked instructions; optional-mask instructions and inherent-mask instructions. Optional-mask instructions include arithmetic operations, logical operations, decisions, and information transfers. When instructions are performed under the control of masks, they usually designate partial words as operands and results. When an optional-mask instruction is performed, the same mask is applied to operands and result. Only those bit positions of the operands which correspond to binary ones in the mask word are used in the operation. The positions of the result location which do not correspond to binary ones in the mask are not altered by the operation. The location of the mask used in an optional-mask instruction is specified by bits 2 through 6 of the command code (see Figure 5-1), in conjunction with bits 2 through 9, 10 through 13, and 19 through 24 of a control register called the mask index register (MXR). A complete description of the way in which the five command code bits, called the partial mask address, are united with the eighteen bits of the MXR to designate the location of the mask will befound in this section under the discussion of the mask index register.

In Mod 8 assembly language, the location of the mask is specified by writing its symbolic tag in the command code field, following the operation code and separated from it by a comma. Thus, the instruction

```
DS , MASK2     WAGES     DEDUCTNS     WEEKSPAY
```

is performed under control of the mask stored in the memory location assigned by the assembler to the symbolic tag MASK2. Since optional-mask instructions use the memory designator bit positions in the partial mask address, it is impossible for these instructions to address control memory, e.g., the addressing forms prohibited are: indirect main memory, direct control register, indexed control register, and indexed indirect main memory. Only direct and indexed main memory addressing are permissible.

The use of masks is not restricted to optional-mask instructions, but extends to the inherent-mask instructions. These instructions, which have the same command code format as the unmasked category (see Figure 5-1), include five shift instructions, a Substitute, and an Extract instruction. The chief distinction between the two types of masked instructions lies in the fact that the inherent-mask instructions use bits from the B-address field rather than from the command code to specify the location of the mask. For the shift instructions,

the low-order six bits of the B-address field are used in conjunction with bits 2 through 9, 10 through 13, and 14 through 18 of the mask index register (MXR) to locate the mask. In the Substitute and Extract instructions, the entire B-address field is used to specify the location of the mask, without reference to the MXR.

A further difference between inherent-mask and optional-mask instructions is that the latter always operate in the "protected" mode; in other words, the portions of the result location corresponding to binary zeros in the mask are preserved during the operation. The inherent-mask group, on the other hand, includes three instructions which operate in the "unprotected" mode, in which the unmasked portions of the result location are cleared to zeros. In Mod 8 assembly language, the location of the mask for a shift instruction is specified in the same way as for optional mask instructions: by writing its symbolic tag in the command code field following the operation code.

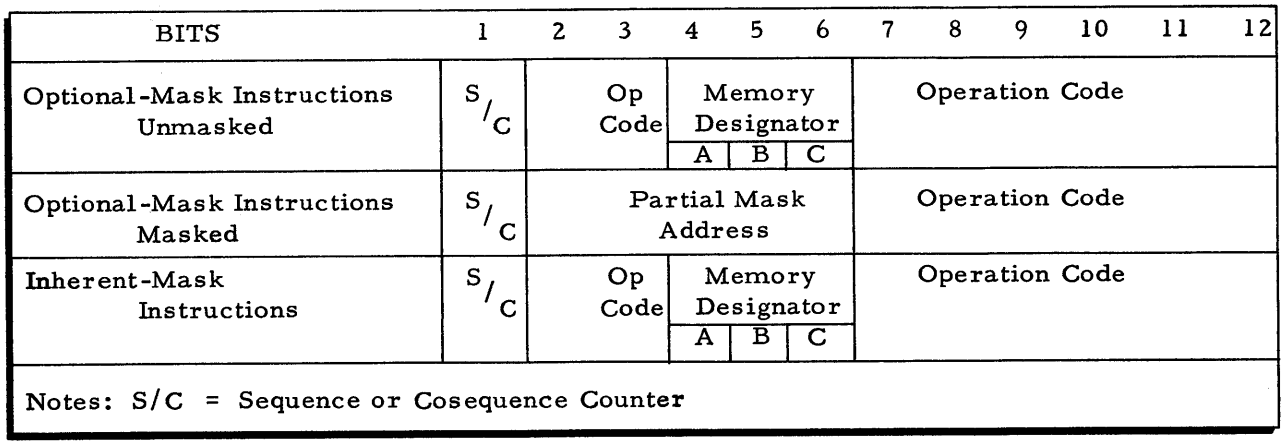


Figure 5-1. Masked Instruction Types

CONTROL REGISTERS

The 32 control registers contained in each group of the word processor's control memory are uniquely designated by nine bits: a 4-bit (0-7) group indicator to specify one of eight control register groups and a 5-bit subaddress (0-31) to designate one of 32 control registers in a group. Table 5-2 lists the 32 registers associated with each group, together with their octal subaddresses and their mnemonic designations in Mod 8 assembly language. The function of each of these registers is detailed in the following discussion.

Each control register has the capacity to store 24 bits of information. The information bits consist of a sign bit ("one" for plus, "zero" for minus) and 23 bits which usually represent an 8-bit array, 4-bit bank indicator, and 11-bit subaddress of a main memory location. When the contents of a control register are modified arithmetically within the control register addition

circuitry, the sign bit determines whether they are incremented or decremented. When a control register word is transferred to the accumulator or to a main memory location, it is stored in the low-order 16 bits (bit positions 33 to 48) of the specified location; in extended mode addressing, it is stored in the low-order 24 bits (bit positions 25 to 48). The high-order 32-bit positions of the location are all cleared to zero; in extended mode addressing, the high-order 24-bit positions are all cleared to zero. Thus, when a control register word is manipulated within the arithmetic unit by an instruction which treats its operands as full-word signed numbers, the word appears to be negative, since it contains four zeros in the sign bit positions. When information is transferred from the accumulator or from a main memory location to control memory, only the low-order 16 bits (the low-order 24 bits in the extended mode) of the word are stored in the control register; the high-order 32 bits (high-order 24 bits in the extended mode) are discarded. Refer to the discussion of addressing (page 5-17) for further detailed explanation.

The discussion of indexed memory location addressing states that whenever a control register is addressed by its full 5-bit subaddress in an instruction address field, it is said to be explicitly addressed; whenever a control register is addressed in any other way, it is said to be implicitly addressed. Instructions using extended addressing must explicitly address the control register to be used for indexing. It is also stated that an index register is implicitly addressed when designated by the 3-bit index register number in an indexed address. Implicit addressing is further illustrated by the following examples:

1. A sequencing counter is addressed implicitly every time an instruction is selected and executed;
2. Two arithmetic control counters are implicitly addressed whenever an N-Word Transfer instruction is executed.

It will be noted that in each of these examples the implicitly addressed control register is called a counter. The following rule may be stated: those control registers designated in Table 5-2 as counters are always automatically incremented (if the control register sign is positive) or decremented (if the control register sign is negative) by one each time they are addressed implicitly; in extended mode, the control registers are incremented or decremented by two each time they are addressed implicitly. When these counters are addressed explicitly in an instruction address field, however, incrementing is not automatic but occurs only under program control, if specified in the address field.

A one in the sign bit of a control register represents plus; a zero represents minus. In general, the value of this bit is changed by explicit addressing. With some control registers, however, the value of the sign bit can be changed by implicit addressing, as discussed under the descriptions of the individual registers.

Table 5-2. Control Register Names, Subaddresses, and Mnemonic Addresses

MNEMONIC ADDRESS	SUBADDRESS (OCTAL)	NAME
AU1	00	AU-CU Counter #1
AU2	01	AU-CU Counter #2
SC	02	Sequence Counter
CSC	03	Cosequence Counter
SH	04	Sequence History
CSH	05	Cosequence History
UTR	06	Unprogrammed Transfer
MXR	07	Mask Index Register
X0	10	Index Register 0
X1	11	Index Register 1
X2	12	Index Register 2
X3	13	Index Register 3
X4	14	Index Register 4
X5	15	Index Register 5
X6	16	Index Register 6
X7	17	Index Register 7
R0	20	General Purpose Register R0
R1	21	General Purpose Register R1
R2	22	General Purpose Register R2
R3	23	General Purpose Register R3
R4	24	General Purpose Register R4
R5	25	General Purpose Register R5
R6	26	General Purpose Register R6
R7	27	General Purpose Register R7
S0	30	General Purpose Register S0
S1	31	General Purpose Register S1
S2	32	General Purpose Register S2
S3	33	General Purpose Register S3
S4	34	I/O Register[1] S4 (PRIVILEDGED)
S5	35	I/O Register[1] S5 (PRIVILEDGED)
S6	36	I/O Register[1] S6 (PRIVILEDGED)
S7	37	I/O Register[1] S7 (PRIVILEDGED)

[1]These registers, in each group, are reserved for central I/O traffic and are not normally usable by the programmer.

Under program control, the contents of a control register may be arithmetically modified within the control register circuitry in one of two ways: by augmenting an index register or by incrementing an explicitly addressed control register. Augmenting an index register does not alter the retained contents of the register, since the augmenter is actually added to the contents of the index register after they have been read out. One and only one address selection occurs each time the index register contents are augmented. Thus, even if the same index register is specified in two successive instructions or twice within the same base address is used for each address selection. Incrementing, on the other hand, alters the retained contents of the control register. After the control register has been selected and its contents used, the increment specified in the address field is actually added to those contents, and the result is returned to the control register before the next address is selected. Thus, when the same control register is addressed twice within an instruction, the contents of the register at the second addressing are different from the contents at the first addressing, unless, of course, a zero increment is specified in the first instance. However, if a sequencing counter is used to address a memory location indirectly in the C address and the instruction is in the same sequencing mode, the contents of the sequencing counter will not be incremented even though an increment is specified in the address field. For example, when the instruction

```
TS   ITEMA   N, R1,   1      N, SC, 5
```

is executed, the word at ITEMA is transferred to the memory location designated by the address stored in control register R1. The contents of R1 are incremented by one after use and replaced in R1. Since the C address in this instruction specifies that the sequence counter be used to address a main memory location, the contents of the sequence counter <u>will not</u> be incremented although an increment of 5 is specified in the C-address field.

Any instruction which can explicitly address a control register may operate on its contents. This means that a control register word may be shifted, may be operated upon arithmetically, may be compared, and may be moved around in either main or control memory. When a control register is brought into the accumulator, bits 1 through 32 of the accumulator are filled with zeros in the normal mode; bits 1 through 24 are filled with zeros in the extended address mode. Since 4 zero bits in the sign position define a negative number, a control memory word is always negative when manipulated in the accumulator, regardless of the value of the control register sign bit.

The peripheral command codes do not provide memory designator bits for explicit addressing of the control memory. It is therefore impossible to read directly into a control register from a peripheral device or to deliver the contents of a control register directly to a peripheral device. For the same reason, it is also impossible to address a control register in an optional-mask instruction.

Each group of control registers forms a control center for a single program. Thus, as many as eight independent programs may be active at the same time. Each program proceeds under control of the sequence or cosequence counter in its own control register group and refers to the other control registers (index registers, mask index register, and so forth) in this group. Direct memory location addressing (normal mode) allows the programmer to address only those memory locations within the area specified by the bank indicator bits of the sequencing counter which selected the instruction. When the main memory is addressed through the control registers, however, the program may have access to any location in memory. Furthermore, when any of the counters in the control memory is incremented, any resulting carry may propagate throughout the full 23-bit address, with the result that the main memory is completely continuous when addressed through these counters. Thus, sequencing of control, reading, writing, and transfer of information may all proceed without regard to bank designation.

In the extended addressing mode, each address field consists of a 23-bit address and a sign bit. Therefore, the main memory is completely continuous when addressed by an extended address instruction, e.g., each address field in an extended address instruction specifies the 11-bit subaddress, the 4-bit bank indicator, and the 8-bit array. The operation is not restricted to any particular area of memory.

Sequence and Cosequence Counters

Each control register group contains two sequencing counters called the sequence counter (SC) and cosequence counter (CSC). Except in the case of Simulator instructions, the programmer may use either of these counters to sequence his program. Furthermore, in any instruction except the Simulator, Proceed, and input/output instructions, he may specify which counter will select the next instruction, with the result that he may change control between the two with complete freedom. The use of two counters in this way is called the bisequence operation mode. Since the behavior of the two counters is identical, the following description of the sequence counter is also applicable to the cosequence counter.

The sequence counter contains a sign and 23 bits which are interpreted by the control circuitry as an array, a bank indicator, and a subaddress. These 23 bits represent the complete address of a main memory location from which an instruction is to be selected. Each time the sequence counter is implicitly addressed for the selection of an instruction, its contents are automatically incremented or decremented by one (according to the value of the sign bit) and immediately replaced in the counter; in extended mode, the contents of the sequence counter are automatically incremented or decremented by two. During the execution of an instruction selected from location N in the normal mode, for example, the sequence counter

contains the quantity N + 1 if the sign bit is positive. In this case, therefore, instructions are taken from successively higher memory locations. If the sign bit is negative, on the other hand, the instructions will be selected from successively lower memory locations.

An instruction that addresses a sequencing counter implicitly by specifying a change in its contents to the location designated by the C-address field an change the sign of the counter. If the C address specifies a direct or indexed memory location address, a positive sign is always inserted in the counter. If, however, the address specified is an indirect memory location or indexed indirect memory location, the new sign of the sequencing counter is determined by the sign of the designated control register, because the complete contents of the control register are transferred to the sequencing counter (refer to "Transfer Instructions" later in this section).

As previously noted, carries may propagate across the entire 23 bits of a sequencing counter during incrementation. Instruction sequences can therefore pass freely from one memory bank to another. If an attempt is made to sequence the counter beyond the assigned memory included for a particular program, however, an attempted memory violation will result and master control is called. The same result will occur if a sequencing counter containing a negative sign and 23 binary zeros is implicitly addressed.

The initial setting of the sequence counter is normally made by transferring from main memory a word whose low-order 24 bits represent the desired sign and the memory address from which the first instruction is to be selected. The loader routine performs this function for programs run under the Mod 8 Operating System. Once the sequence counter is set and addressed, it continues to select instructions from successive locations until an instruction is executed which specifies the alternate counter as the source of the next instruction or which changes the contents of the counter itself through explicit or implicit addressing. When an instruction specifies a change of counter but not a change in the contents of a counter, the only change which occurs in the two sequencing counters is the normal incrementation of the counter which selected the instruction. An instruction which explicitly addresses a sequencing counter as a result location simply causes the contents of that counter to be replaced. For example, the instruction

TX Z,AU1 - Z, SC

merely replaces the contents of the sequence counter with the contents of AU1, so that the next instruction is selected from the location whose address is stored in AU1. No record of such a sequence change is retained by the machine. An instruction which changes the contents of a counter by implicit reference, on the other hand, alters the contents of the counter specified as the source of the next instruction and stores the contents of the counter which selected this

instruction in the history register (see below) associated with the counter whose contents are changed. Thus a record of the last implicit sequence change which affected the contents of either counter is always available internally. For example, an instruction selected under control of the sequence counter specifies the cosequence counter as the source of the next instruction and directs the program to transfer a word from A to B and select the next instruction from the location specified by C:

```
TS   C   RECORD   OUTPUT   SECTIONA
```

This instruction puts the address tagged SECTIONA in the cosequence counter, where it is selected as the address of the next instruction, and stores the incremented contents of the sequence counter in the cosequence history register.

History Registers

For each sequencing counter in the system, there is a corresponding history register called the sequence history register (SH) or the cosequence history register (CSH). These registers are used to store the entire contents of a sequencing counter (including sign) whenever the counter is implicitly addressed by an instruction specifying a change in its contents. If an instruction selected by the sequence counter from location M specifies a sequence change to location N, and the next instruction is also to be selected by this counter, then the address M + 1 is stored in the sequence history register and the sequence counter itself is set to the address N. If, however, the alternate counter is specified as the source of the next instruction, then the cosequence counter is set to N and M + 1 is stored in the cosequence history register. In other words, unless they are altered by explicit addressing, the contents of a history register always represent the incremented address of the instruction which last changed the contents of the associated sequencing counter by implicit reference. As previously noted, no change in the history register occurs if the contents of a sequencing counter are changed by explicit addressing.

Index Registers

Each of the word processor's control register groups contains eight (designated X0-X7) index registers. Like the other control registers, they contain 24 bits normally interpreted as a sign and a main memory or control register address. Although the index registers must always be loaded and unloaded by the use of an explicit address, they are always implicitly addressed by a 3-bit number when used for their intended purpose in indexed addressing. As explained later in this section, in the normal mode an indexed address field includes a 3-bit index register number and an 8-bit augmenter to be added to the low-order contents of this register. Instructions using extended addressing must explicitly address the control register to be used for indexing. The indexed address field in the extended mode includes a 5-bit control register subaddress and a 17-bit augmenter to be added to the low-order contents of this

register. The retained contents themselves are not modified; the control register addition circuitry merely uses the contents, together with the augmenter, to generate the main memory or control register address of an operand or result location. Since the sign of the register may be positive or negative, at the programmer's option, the generated address may be higher or lower than the base address stored in the register. The word processor accepts augmenters valued from 0 to 255 (or 0 to 131,071 in the extended mode).

Mask Index Register

Each control register group contains a mask index register (MXR) which is implicitly addressed whenever an optional-mask instruction or a shift instruction is executed. The 24 bits of this register are interpreted as a sign, an 8-bit array, a bank indicator, and the high-order portions of two different subaddresses. Bits 14 through 18 specify a partial address for masks used with the shift instructions; bits 19 through 24 serve the same purpose for masks used in optional-mask instructions (see Figure 5-2).

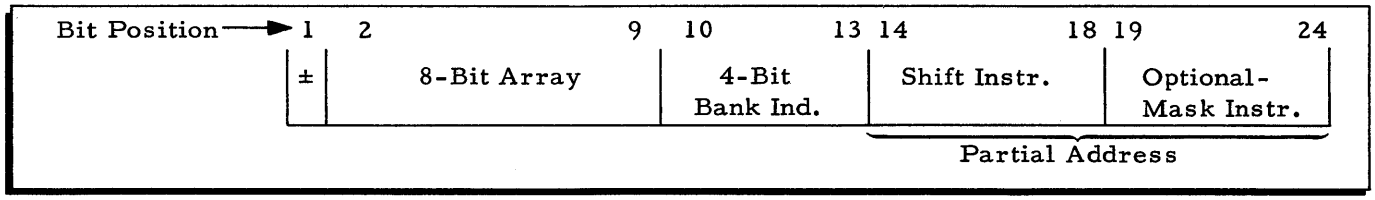


Figure 5-2. Mask Index Register

The low-order portions of the mask addresses are found in the instructions themselves. When a shift instruction is executed, the word processor unites the low-order six bits of the B-address field with the array bits, bank indicator, and bits 14 through 18 from the mask index register to form the complete main memory address of the mask. This process is illustrated in Figure 5-3.

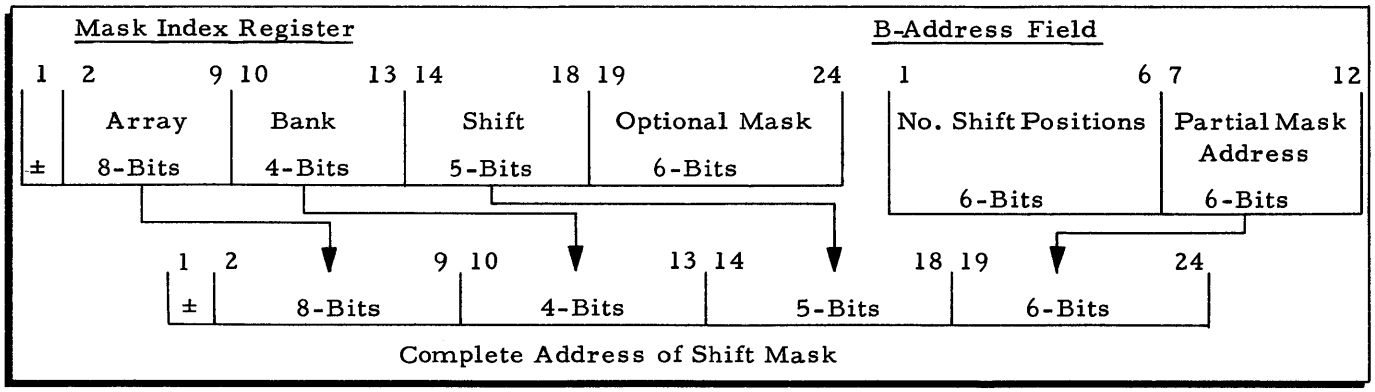


Figure 5-3. Generated Mask Address in Shift Instructions

When an optional-mask instruction is executed, the word processor attaches the 5-bit partial mask address from the instruction command code to the array bits, bank indicator, and bits 19 through 24 from the mask index register to form the complete main memory address of the mask. This process is illustrated in Figure 5-4.

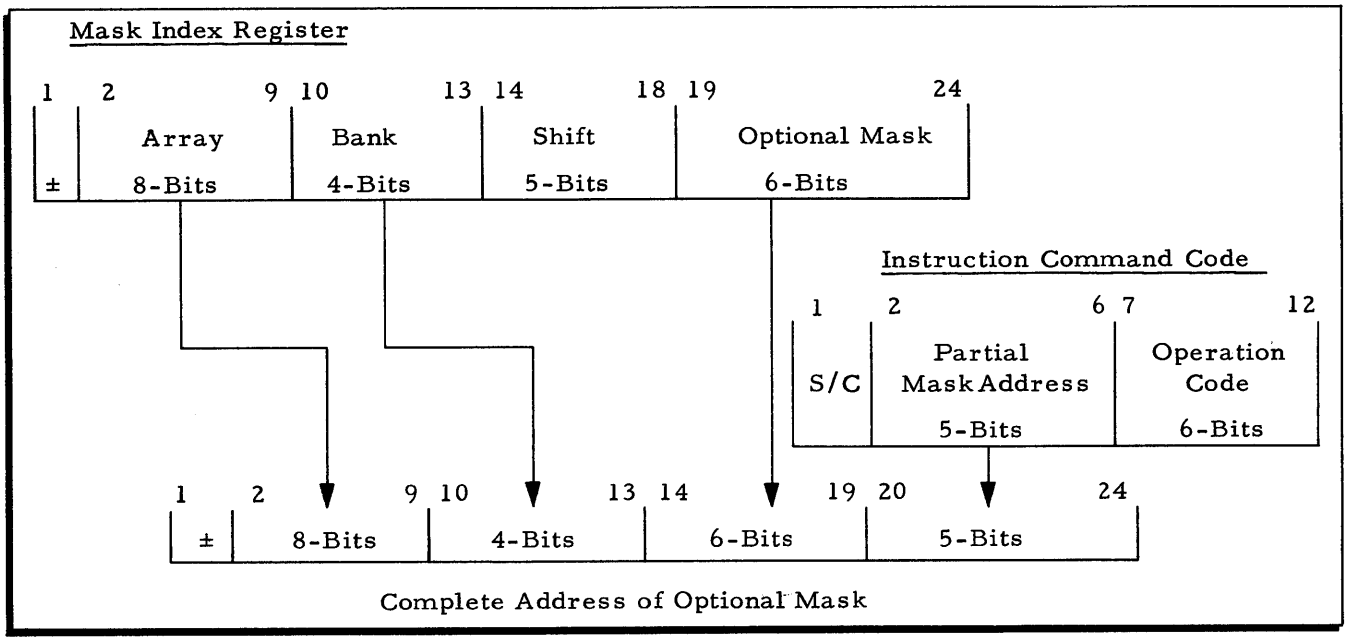


Figure 5-4. Generated Mask Address in Optional-Mask Instructions

Since the mask index register contains array and bank indicator bits, both shift masks and optional masks may be stored in any location of a program's assigned memory area. The value of the sign bit is not relevant in locating masks.

The mask index register is set by explicit addressing. Each time the programmer loads the register, he designates 96 memory locations as mask addresses, 64 for shift masks and 32 for optional masks. Since the programmer may change the contents of the mask index register whenever he wishes, the number of masks available for his use is virtually unlimited.

General Purpose Registers

Each control register group contains 12 general purpose registers (R0-R7, S0-S3). Like the index registers, general purpose registers are used primarily for address modification. Their use differs from that of index registers, however, in several important respects. First, they are always addressed explicitly. Secondly, the specified increment, which has an upper limit of 31 in the normal mode and 131,071 in the extended mode, alters the retained contents of the register after use. As in the case of index registers, the address of a memory location generated by adding the increment to the original contents of the register may be

higher or lower than the address originally contained in the register, according to the value of the sign bit. These registers are used mainly in the indirect addressing mode to address an operand or a result location in assigned memory, but they may also be used as programmed counters, as temporary storage for the contents of other control registers, and for any other purpose the programmer may devise.

Arithmetic Control Counters

Each control register group contains two arithemetic control counters (AU-CU counters) known mnemonically as AU1 and AU2. One or both of these counters are implicitly addressed, loaded, and automatically incremented during execution of an N-Word Transfer, Item Transfer, Record Transfer, or Simulator instruction. As an example, during execution of an N-Word Transfer of 10 words, the initial setting of AU1 represents the main memory or control register address from which the first word is to be transferred, while the initial contents of AU2 represent the location to which this word will be delivered. As successive words are transferred, the counters are automatically incremented to specify a source and result address for each word transferred. At the completion of the instruction, the counters contain addresses equal to their initial settings plus ten.

Implicit reference to an arithmetic control counter always causes the sign bit in the counter to be made positive with the following execption:

> If an arithmetic control counter is loaded with the contents of a control register, its sign bit takes the value of the sign stored in the control register.

Since carries may propagate across the low-order 23 bits of the counter, a record which is divided between two memory banks may be transferred as easily as one contained entirely in one bank. It should be noted, however, that when the contents of an arithmetic control counter are interpreted as a control register address, a subaddress overflow will not change the group indicator but instead will change the value of the tabular bit from zero to one.

Like other control registers, the arithmetic control counters may be addressed explicitly in order to transfer their contents to main memory or to use them as general purpose registers. The programmer who uses them thus, of course, must remember that information stored in one or both of these registers will be destroyed by the execution of certain instructions. Whenever these control registers are addressed explicitly, they lose their identity as automatically incremented counters. The arithmetic control counters are described more fully in connection with the instructions which use them.

Unprogrammed Transfer (Internal Interrupt) Register

The word processor is so designed that the occurrence of certain unusual events during execution of a program does not stop the machine but rather an internal interrupt occurs which effects a transfer of control out of the normal sequence of the program to initiate appropriate action as specified in a programmed subroutine. The unprogrammed transfer register (UTR) in the control register group controlling the program is the source to the location of these subroutines designed to handle the seven different types of conditions which may cause an unprogrammed transfer.

The unprogrammed transfer register is initially loaded, by explicit addressing, with 24 bits which represent a sign, an 8-bit array, a bank indicator, and an 11-bit main memory subaddress. The address thus loaded, called U, must be even or an error will result and master control will be called when an unprogrammed transfer is attempted. When an internal interrupt occurs, the control circuitry reads out the contents of the UTR and inserts a one into bit 24 if the instruction causing the unprogrammed transfer was selected from the cosequence counter. The instruction itself is stored in the address thus generated. In other words, the instruction causing the transfer will be stored in U if it was selected by the sequence counter or in U + 1 if it was selected by the cosequence counter.

The conditions which result in internal interrupts are listed in Table 5-3. Each of these events causes the execution of one instruction whose address is formed by adding a constant (n) to the contents of the UTR, under control of the UTR sign bit. If the UTR contains a positive sign, the unprogrammed transfer causes execution of an instruction whose address is U + n. If the sign bit in the UTR is negative, the unprogrammed transfer is made to U - n.[1] If the instruction causing the transfer was selected by the sequence counter, n is an even number from 2 to 14; if it was selected by the cosequence counter, n is an odd number from 3 to 15. Thus, the value of n depends upon the event that caused the transfer and the counter that selected the instruction, as illustrated in Table 5-3.

It should be noted that every direct memory location address in an instruction selected by the unprogrammed transfer register has the array and bank indicator bits from the UTR appended to it to form a complete 23-bit address. In particular, if a change of sequence is made by direct main memory addressing in such an instruction, the bank indicator bits from the UTR are placed in the sequence or cosequence counter.

[1] In Table 5-3, the unprogrammed transfer addresses are listed as U ± n. For the sake of simplicity, they are referred to as U + n throughout the section.

Table 5-3. Unprogrammed Transfers of Control

Event Causing Unprogrammed Transfer (Internal Interrupt)	First Word of Instruction Stored In	Next Instruction From
Parity Failure		
Sequence Counter	U	U ± 2
Cosequence Counter	U + 1	U ± 3
Beginning or End of Tape[1]		
Sequence Counter	U	U ± 4
Cosequence Counter	U + 1	U ± 5
Read or Write Error[1]		
Sequence Counter	U	U ± 6
Cosequence Counter	U + 1	U ± 7
Addition or Subtraction Overflow		
Sequence Counter	U	U ± 8
Cosequence Counter	U + 1	U ± 9
Division Overcapacity		
Sequence Counter	U	U ± 10
Cosequence Counter	U + 1	U ± 11
Exponential Underflow		
Sequence Counter	U	U ± 12
Cosequence Counter	U + 1	U ± 13
Exponential Overflow		
Sequence Counter	U	U ± 14
Cosequence Counter	U + 1	U ± 15

[1] Special I/O interrupt only effective with the Read Forward (RF) and Write Forward (WF) instructions.

The execution of an unprogrammed transfer does not alter the contents of either sequencing counter. Thus, control returns immediately to the normal sequencing of the program unless the unprogrammed transfer instruction itself changes the contents of the sequencing counter from which the next instruction is selected. Note, however, that the bisequence bit in the instruction causing the transfer is ignored; therefore, no change in sequencing counters will occur.

By inserting the appropriate address into the unprogrammed transfer (internal interrupt) register, the programmer may select any area of memory for use as a corrective routine selection table. He may also change the contents of the UTR at any point in the program or he may change the contents of any entry in the table to correspond with the particular portion of the program being executed.

With the exception of the parity failure condition, the events resulting in internal interrupts are associated with a few specific instructions. Further details on these conditions

appear under the descriptions of the appropriate instructions. Parity failures are discussed in detail in the paragraphs which follow.

If a parity error is detected in a word selected from memory, the current instruction is completed and a parity error unprogrammed transfer is executed to U + 2 or U + 3, except in the following cases:

1. If the word selected is the next instruction to be performed, a master control call occurs.
2. If the word selected is a control memory word, parity failure results in a master control call.
3. If a parity failure occurs during data transfer in a Record or Item Transfer, the instruction is not completed, and the unprogrammed transfer to U + 2 or U + 3 occurs immediately.
4. If a parity failure occurs in a word being transferred from a peripheral device under control of a Read instruction, a read error indication is stored, and the unprogrammed transfer action is as specified under the discussion of read errors in Section XI of the Model 800 or 1800 Programmers' Reference Manuals.
5. If a parity failure occurs in a Check Parity instruction, no unprogrammed transfer takes place, but a sequence change occurs as described in Section XII of the Model 800 or 1800 Programmers' Reference Manuals.
6. During a Compute Orthocount instruction, parity failure is ignored, and no unprogrammed transfers occur. Refer to Section XII of the Model 800 or 1800 Programmers' Reference Manuals.

If a parity failure unprogrammed transfer occurs as the result of executing an instruction that normally changes the contents of the sequencing counter to the location specified by the C-address field, the sequencing counter is not changed on completion of the instruction.

ADDRESSING

The memory subsystem of the Honeywell 8200 word processor consists of from two to eight modules of core memory and a memory controller. Each module is four characters in width and generally 32,768 four-character groups in length. Data storage capacities of the memory subsystem range from 262,144 to 1,048,576 characters (32,768 to 131,072 words). Each main memory location is directly addressable and is designated by a 24-bit configuration. This configuration of bits may also be thought of as a sign bit, an 8-bit array which provides additional direct accessibility to memory locations throughout the system, a 4-bit bank indicator to specify a memory bank, and an 11-bit subaddress to specify, in binary, one of the 2048 locations in a bank. The bit structure of a main memory address is shown in Figure 5-5. (It is sometimes convenient to express the value of such a 24-bit configuration as eight octal digits. In this notation, the memory addresses of 131,072 word system range from 00000000 to 37777777.)

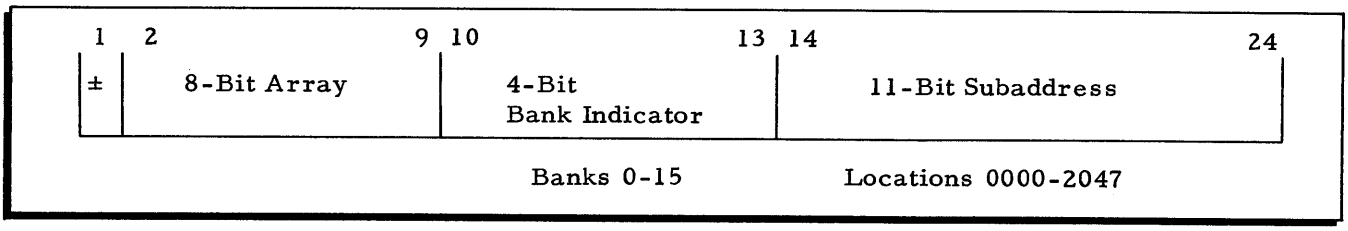


Figure 5-5. Main Memory Address

The control memory consists of 256 control registers divided into eight groups of 32 registers each. Every control register is directly addressable by a unique 9-bit configuration. This array of bits may also be thought of as a 4-bit group indicator designating one of the eight control register groups and a 5-bit subaddress specifying one of the 32 registers in a group (see Figure 5-6).

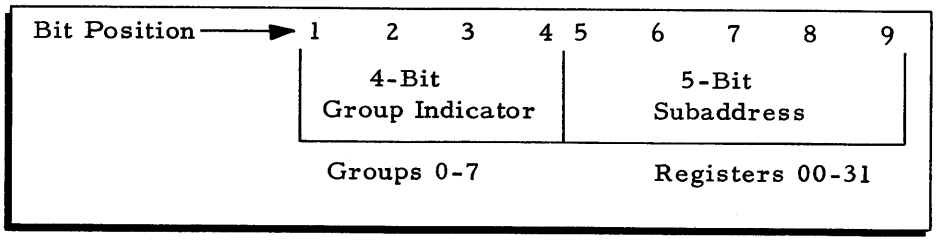


Figure 5-6. Control Register Address

As described in Section II, the word processor's instruction format normally consists of four 12-bit fields: the command code, A address, B address, and C address. Since both the main and the control memories are directly addressable, some means must be provided to specify which memory is being addressed in each of the three address fields of an instruction. This is accomplished by defining one memory designator bit position in the command code of the instruction for each of the three address fields. A zero designator bit indicates that the respective address refers to main memory; a designator bit of one indicates that the address refers to a control memory location (control register). For those command codes which do not provide the memory designator bit positions, designators of zero are always implied and the respective addresses are always interpreted as main memory addresses.

Since an instruction address field includes 12 bits (plus a memory designator bit in the command code, explicit or implied), it does not precisely specify a complete main memory address (23 bits) or a complete control memory address (9 bits). The instruction address bits may be interpreted by the word processor in a number of ways to form a complete main or control memory address. For example, direct addressing is the explicit statement of the desired main or control memory subaddress in the instruction. Indexed addressing refers to the technique of augmenting a main or control memory address stored in an index register

to form the desired address. Indirect addressing refers to the technique of stating the address of a control register in which the desired main memory address is stored. Main memory locations can be addressed in any of these ways; control registers are addressed either directly or by indexing.

In addition to the above, the word processor also has an extended addressing format which allows direct addressing of the entire memory, regardless of its size. This format addressing utilizes an extended instruction format which occupies two consecutive 48-bit words: a 24-bit command code and three 24-bit address fields. Employment of the double length address field allows direct accessibility to any of the 131,072 words in main memory. Extended addressing is also provided for all the other forms of addressing. For a discussion of the formatting of instructions in both the normal and extended mode, refer to Section II.

Direct Memory Location Address (Normal and Extended)

Each address field in the normal mode instruction word consists of 12 binary digits. Bit 1 of this 12-bit configuration specifies whether the address is direct or indexed. If bit 1 is zero, the address is direct; if bit 1 is one, the address is indexed.

If bit 1 of an address field is zero and the corresponding memory designator bit is zero (main memory), then the remaining 11 bits of the address field are interpreted as a subaddress designating one of the 2048 locations in a bank of memory. The bank indicator and array bits stored in the sequencing counter[1] which selected the instruction are appended to this subaddress to form a complete 24-bit address (see Figure 5-7). Thus, with the appropriate setting of the sequencing counters, a program may operate directly within any bank throughout the entire memory.

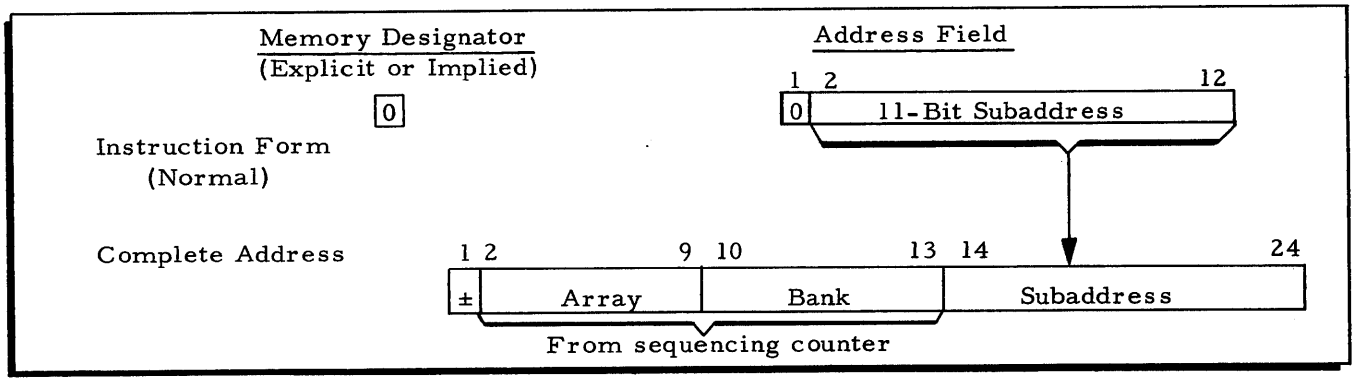


Figure 5-7. Normal Direct Memory Location Addressing

[1] Although instructions are normally selected by a sequence or cosequence counter, they may be selected by an unprogrammed transfer register. In this case, the bank indicator and array bits are taken from the unprogrammed transfer register to form a direct memory location address.

In extended mode, if the high-order bit of the address field and the corresponding memory designator bit are both zeros, then the low-order 23 bits of the address field are interpreted directly as an extended main memory address (see Figure 5-8). This extended form of addressing allows direct access to any location throughout the amount of memory in the system.

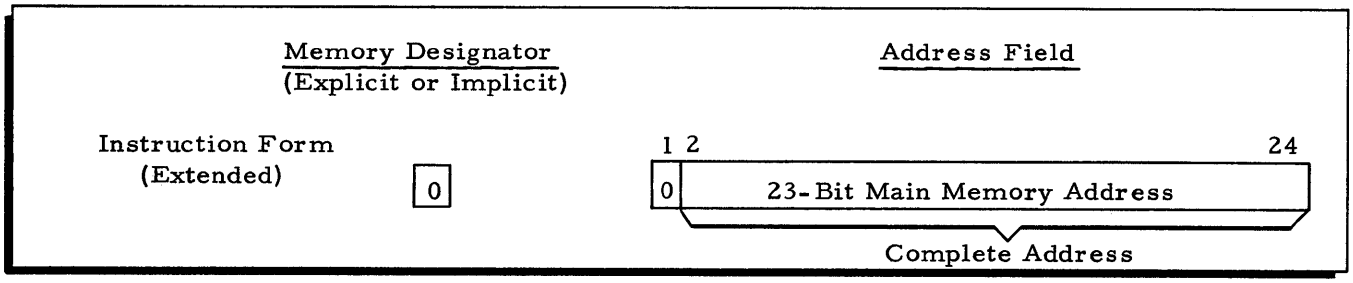


Figure 5-8. Extended Direct Memory Location Addressing

In a Mod 8 assembly instruction, a direct memory location address is specified by a symbolic tag or by address arithmetic, in which the address is designated according to its relative position with reference to the instruction in which it appears or with reference to a symbolic tag. These three types of direct memory location addressing are illustrated in the assembly instruction:

```
DS    ENTRY    C, +20    ENTRY    -2
```

ENTRY is the symbolic tag of a main memory location; C, +20 represents the location 20 after the location of the instruction itself; and ENTRY -2 represents the location two before that tagged ENTRY.

Direct Control Register Address (Normal and Extended)

In the normal mode, if bit 1 of the address field is zero (direct) and the memory designator bit is one (control memory), bits 8 through 12 of the address field are interpreted as the subaddress of one of the 32 control registers in the group which includes the sequencing counter that selected the instruction. The word processor attaches to this subaddress the group indicator associated with the sequencing counter to form a complete 9-bit control register address (see Figure 5-9). If bit 7 of the address field (called the tabular bit) is zero, then the 9-bit array is interpreted as a direct control register; that is, the specified register is used as an operand location or as a result location. Bits 2 through 6 of the address field specify an increment in the range 0 through 31 (in binary), which may be added, under control of the control register sign bit, to the low-order bits of the control register after use, thereby altering them permanently. If the control register sign bit is positive, the value of the increment is added to the contents of the control register, and the contents are said to be incremented. If the sign is negative, the value of the increment is subtracted from the contents of the control register and the contents are said to be decremented. Incrementing (or decrementing)

always occurs when the control register is addressed as the source of an operand, never when the control register is addressed as a result location. The formation of a direct control register address is illustrated in Figure 5-9.

Since the group indicator attached to the control register subaddress is always that of the sequencing counter which selected the instruction, a direct control register address always refers to the control register group containing the sequencing counter which selected the instruction.

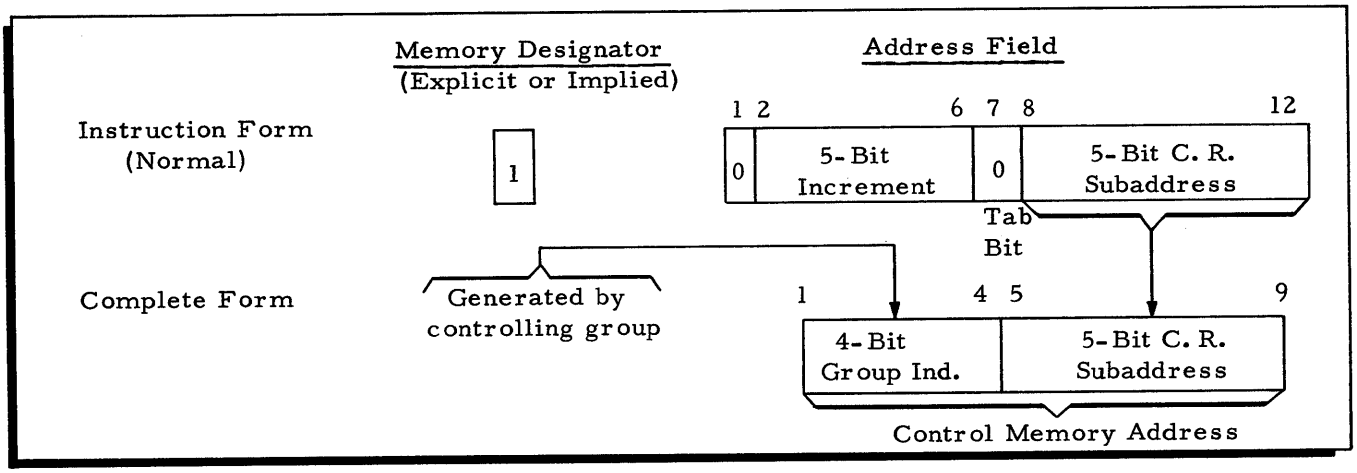


Figure 5-9. Normal Direct Control Register Addressing

If the specified control register is an operand location, the low-order 15 bits of its contents and the sign bit are transferred to the low-order 16 bits, respectively, of main memory or the accumulator. Incrementing occurs after use; thus, the contents of the specified register are permanently altered (see Figure 5-10).

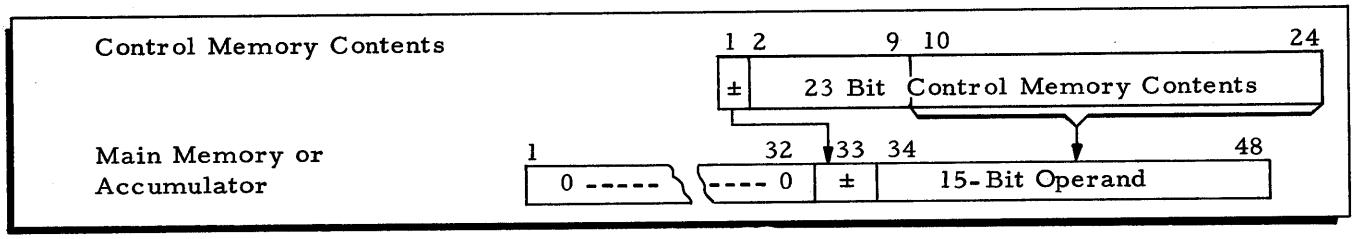


Figure 5-10. Operand Location in Normal Mode

If, however, the specified control register is a result location, the low-order 15 bits plus the sign bit are transferred respectively, from the low-order 16 bits of the accumulator to the low-order 15 bits and the high-order bit positions of the specified control register, together with the bank and array bits from the sequencing counter. No incrementing takes place (refer to Figure 5-11).

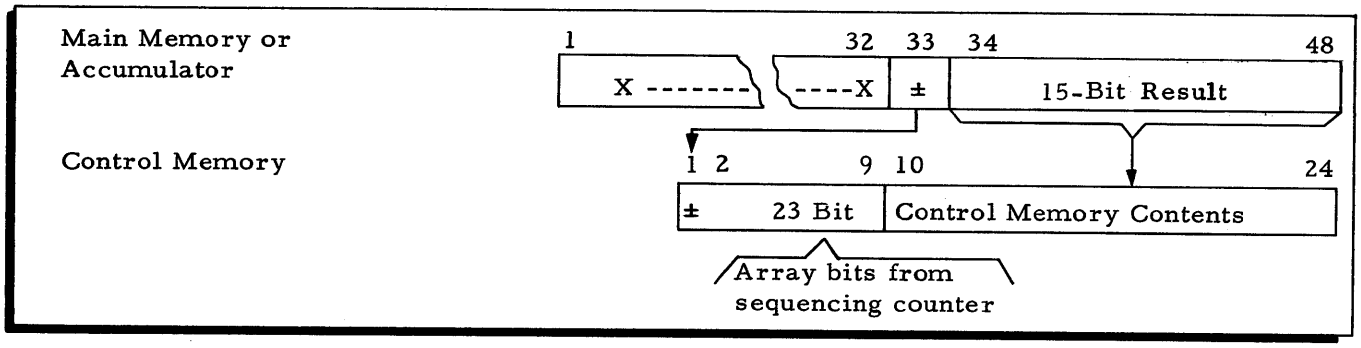


Figure 5-11. Result Location in Normal Mode

If, in the extended mode, the high-order bit of the address field is a zero, the corresponding designator bit is a one, and the tabular bit is a zero, then the control memory address, specified by the low-order five bits of the address field and the controlling group, designates a control memory operand location or a control memory destination for a result delivery. Figure 5-12 shows the results generated in extended direct control register addressing and Figure 5-13 illustrates an operand location in the extended mode.

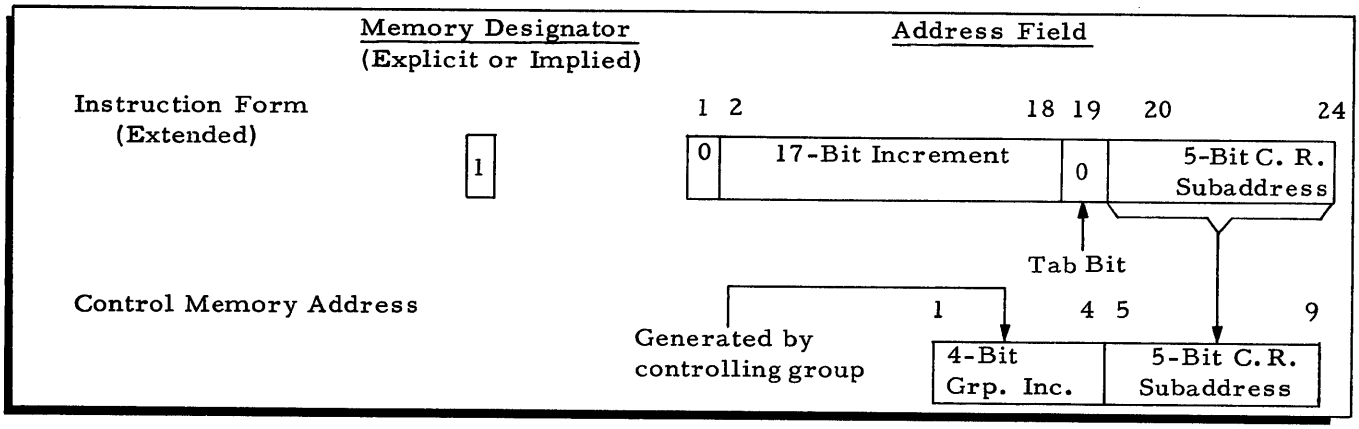


Figure 5-12. Extended Direct Control Register Addressing

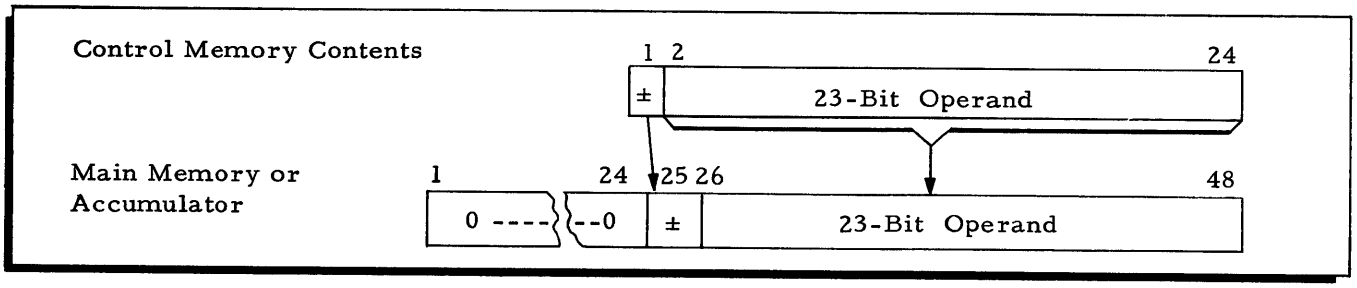


Figure 5-13. Operand Location in Extended Mode

If the control register specified is an operand, its contents will be incremented, after use, by the value of bits 2 through 18 of the address field (maximum of 377777 octal or

131,072 decimal). If the specified control register is a result location (see Figure 5-14), no incrementing takes place. Control register operands are always considered to contain 23 bits plus a sign bit for both operands and results.

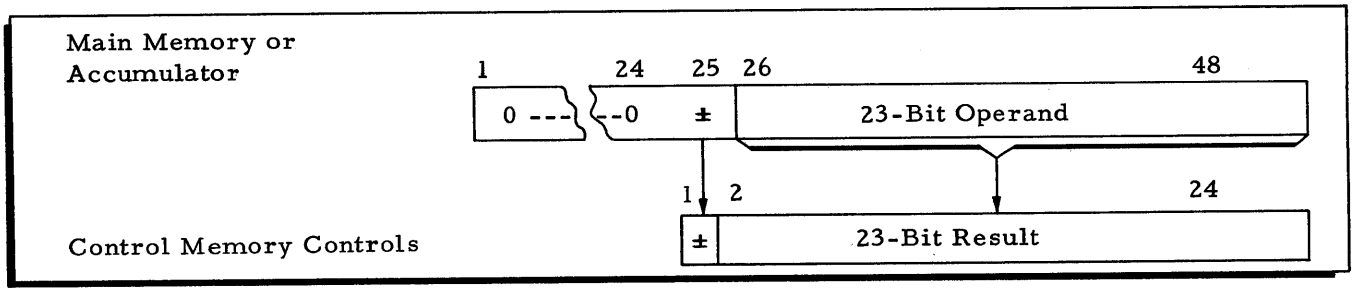


Figure 5-14. Result Location in Extended Mode

In Mod 8 assembly language, the direct address of a control register is indicated by the letter Z, followed by a control register designation and an unsigned increment from 0 to 31 (extended mode from 0 to 131,071), all separated by commas. The letter Z, in effect, represents a one in the memory designator bit position of the command code, a zero in the first bit position of the address field, and a zero in the tabular bit position of the address field. The control register designation may be either the numeric subaddress or the mnemonic subaddress of the desired register, as shown in Table 5-2. If the contents of the control register are not to be altered, the programmer may specify an increment of zero or may omit the increment entirely. The assembly address

```
Z,   R1,   10
```

indicates that general purpose register R1 is addressed directly as the source of an operand or as a result location; if addressed as an operand source, its contents are to be incremented (or decremented) by 10 after use. The address

```
Z,   X0
```

indicates that index register 0 is directly addressed and that no incrementing is to take place.

It should be noted that the relative positions of the control register subaddress and the increment are reversed in assembly language from their machine language arrangement. In other words, the increment appears in the low-order position of the assembly address, but in the high-order bits of the machine address field.

Indexed Memory Location Address (Normal and Extended)

Each control register group includes eight index registers. An indexed address refers to one of these registers in the control register group of the sequencing counter which selected the instruction and is defined by a one in bit 1 of the address field. The remaining 11 bits of the address field are interpreted as an index register number and an augment to be added to

the contents of that index register before use. Bits 2 through 4 designate one of the eight registers in the group, while bits 5 through 12 specify, in binary, a number from 0 to 255 which augments the low-order bits of the contents of the index register. (Note that an indexed address specifying index register 7 with an augmenter of 255 is interpreted as an inactive address; therefore, index register 7 is limited to an augment of 254 maximum. See page 5-11.) It should be emphasized that in indexed addressing the index register is not identified by its 5-bit subaddress, but by only three bits which designate its position within the group of eight index registers. Whenever a control register is addressed in an instruction by its full 5-bit subaddress, it is said to be explicitly addressed. When it is addressed in any other way, it is said to be implicitly addressed. Since the index register in an indexed address is denoted by only three bits, an index register is always addressed implicitly in indexed addressing.

Like all control registers, an index register has the capacity to store 24 information bits of which bit 1 is a sign bit. If the memory designator bit in the command code of the instruction is zero (either explicit or implied), the low-order 23 bits stored in the specified index register are interpreted as a bank indicator, array bits, and an 11-bit main memory subaddress. When the instruction is performed, the 8-bit augment is added to the stored 23-bit address, under control of the index register sign, to form the desired main memory address. This process has no effect upon the contents of the index register, which retains the unaugmented address. The interpretation of an indexed memory location address (normal mode) is shown in Figure 5-15.

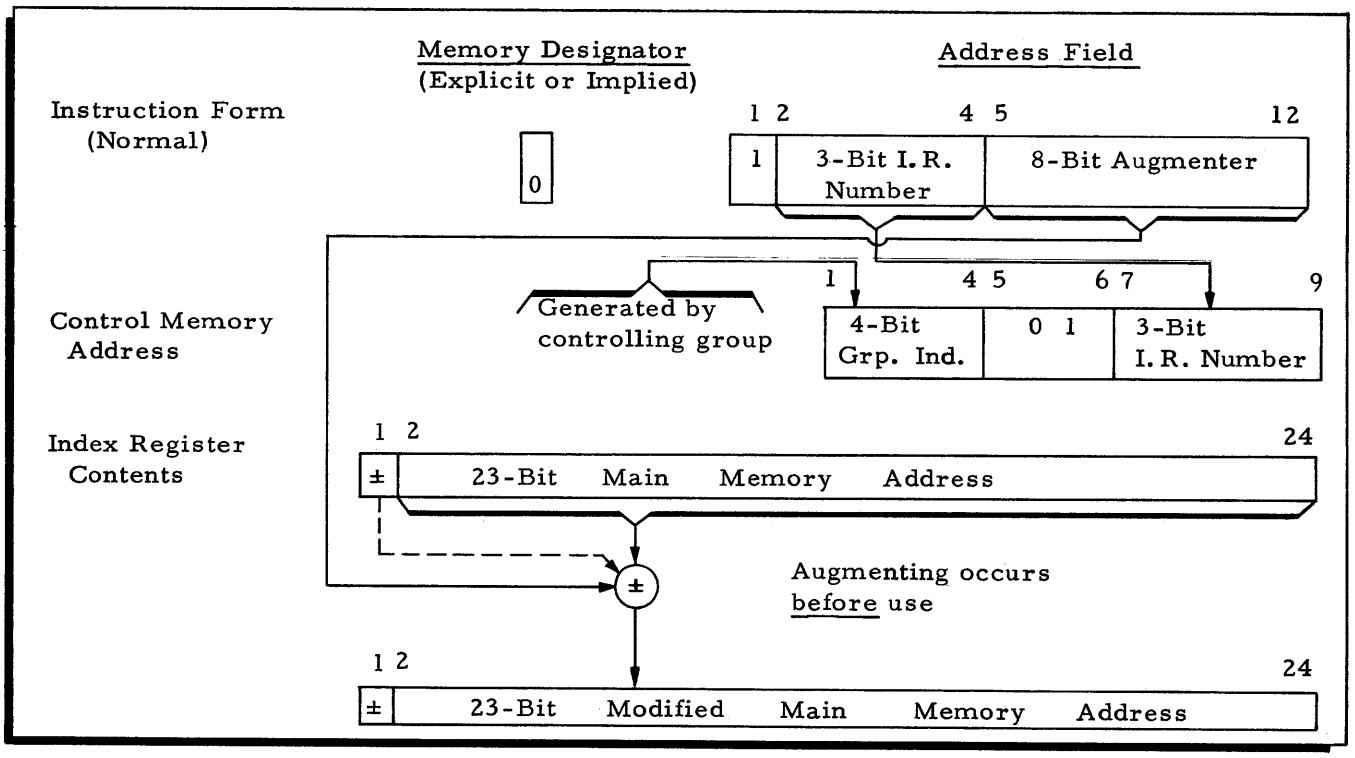


Figure 5-15. Normal Indexed Memory Location Addressing

Since the index register contains a full 24-bit memory address, indexed addressing, unlike direct addressing, permits the programmer to address locations in any main memory bank, regardless of the bank indicator stored in the controlling sequencing counter. This type of addressing is also useful in processing multiword items or in referring to a stored table, where the address of the first word of the item or table is stored in an index register and all references to the item or table are made using the index register with appropriate augment. It must be remembered, however, that positive augmentation occurs only if the index register sign is positive. If the sum of the augment plus the stored subaddress exceeds 2047, a carry occurs into the bank indicator, and the resulting address designates a location in a different bank from the address stored in the index register.

The extended form of addressing allows any control register in the controlling group to be used as an index register. If the high-order bit of the address field is a one, and both the corresponding memory designator bit and the tabular bit are zeros, then the contents of the control register, as specified by the 5 low-order bits of the address field and the controlling group, are interpreted as a signed 23-bit main memory address. The address is augmented before use by the value of bits 2 through 18 of the address field to form a complete 24-bit main memory address. The contents of the specified control register remain unaltered. Figure 5-16 shows the extended indexed memory location addressing.

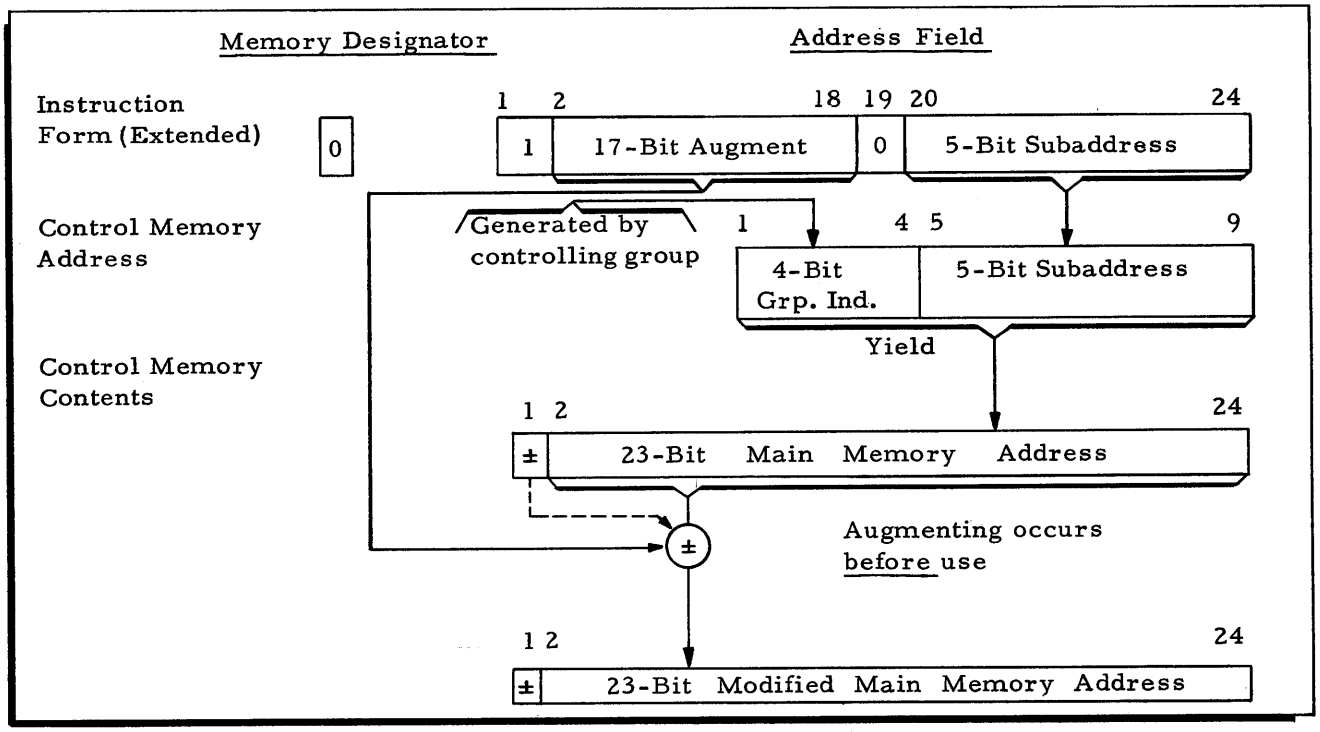


Figure 5-16. Extended Indexed Memory Location Addressing

In Mod 8 assembly language, an indexed memory location address is indicated by writing an index register number (from 0 to 7) followed by a comma and a number from 0 to 255 or a symbolic tag to represent the augmenter. Thus the address

5, 10

specifies that the contents of index register 5 in the related control register group are to be augmented by 10 to form the complete memory address of an operand or result location.

Indexed Control Register Address (Normal and Extended)

In the normal mode, if bit 1 of the address field is one (indexed) and the memory designator bit is one (control memory), the address field is interpreted as an index register number and an augment. The augmented contents of the specified index register are then interpreted as a control register address rather than a main memory address. When the augment has been added to the low-order eight bits of the index register, the resulting configuration is interpreted as shown in Figure 5-17.

Two facts illustrated by Figure 5-17 should be particularly noted. Since the augmented contents of the index register are interpreted as a control register address complete with group indicator, this type of addressing, unlike direct control register addressing, permits the programmer to address control registers in any group. As noted earlier, this facility has particular significance in connection with access to certain control registers involved in reading and writing operations.

The second point involves the value of the tabular bit. Depending upon the original contents of the index register and the value of the augmenter, the tabular bit in the modified index register contents may be zero or one. If this bit is zero, then the control memory address generated from there denotes the location of a control memory operand or a control memory result. Operand and result locations in the normal mode are then treated as described under the discussion of direct control register addressing (Figures 5-10 and 5-11). If the tabular bit is one, on the other hand, the type of addressing is indirect, as described below under the discussion of indirect addressing. Regardless of the value of the tabular bit, the contents of the control register will be permanently modified, after use, by the value of the 5-bit increment, under control of the sign of the control register itself provided that the control register is not addressed as a result location. As always, the contents of the index register are not altered by the indexing process.

In the extended mode, if the memory designator bit and the high-order bit of the address field are both ones, then the contents of the control register, as specified by the controlling group and the five low-order bits of the address field, are interpreted as shown in Figure 5-18.

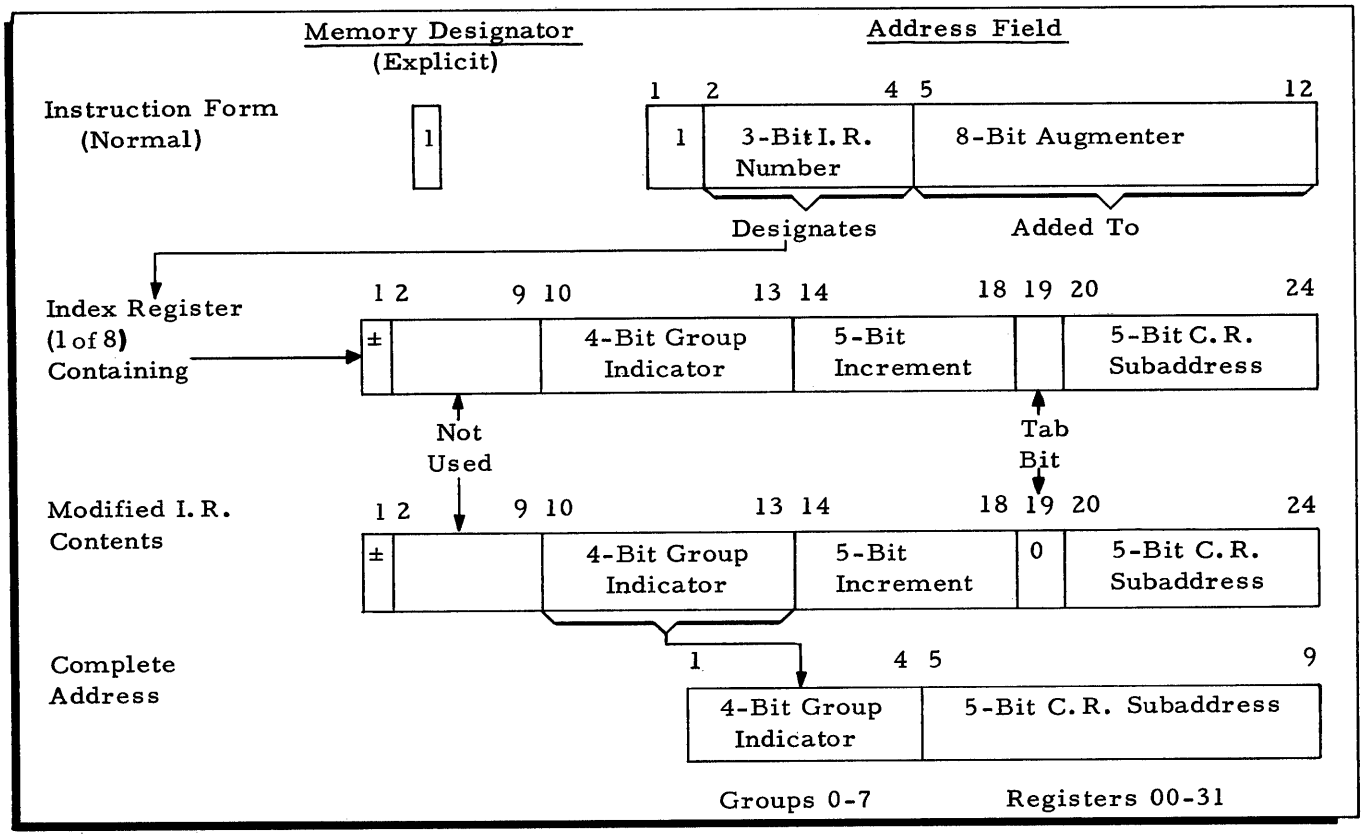


Figure 5-17. Normal Indexed Control Register Addressing

The control register being addressed is then augmented by the value of bits 2 through 18 of the address field. The modified control register is then interpreted in the normal manner for indexed control register addressing, except that bit 1 is the sign bit, control memory operands are 23 bits plus a sign bit, and the group number is always 4 bits (bits 10-13). Note that master control is designated by 1000_2 and master control auxiliary is designated by 1001_2. Addresses above 1001_2 are illegal and will result in a call to master control.

Operand and result locations in the extended mode are treated as described under the discussion of direct control register addressing (Figures 5-13 and 5-14 illustrate the extended mode operand and result locations).

In Mod 8 assembly language, an indexed control register address takes the form

Index Register Designator, Z, Control Register Designator, Increment

The index register designator is a number from 0 to 7 which specifies one of the eight index registers related to the controlling sequencing counter. The control register designator may be a number from 0 to 31 or it may be mnemonic (see Table 5-2). The increment may be a number from 0 to 3 or it may be omitted.

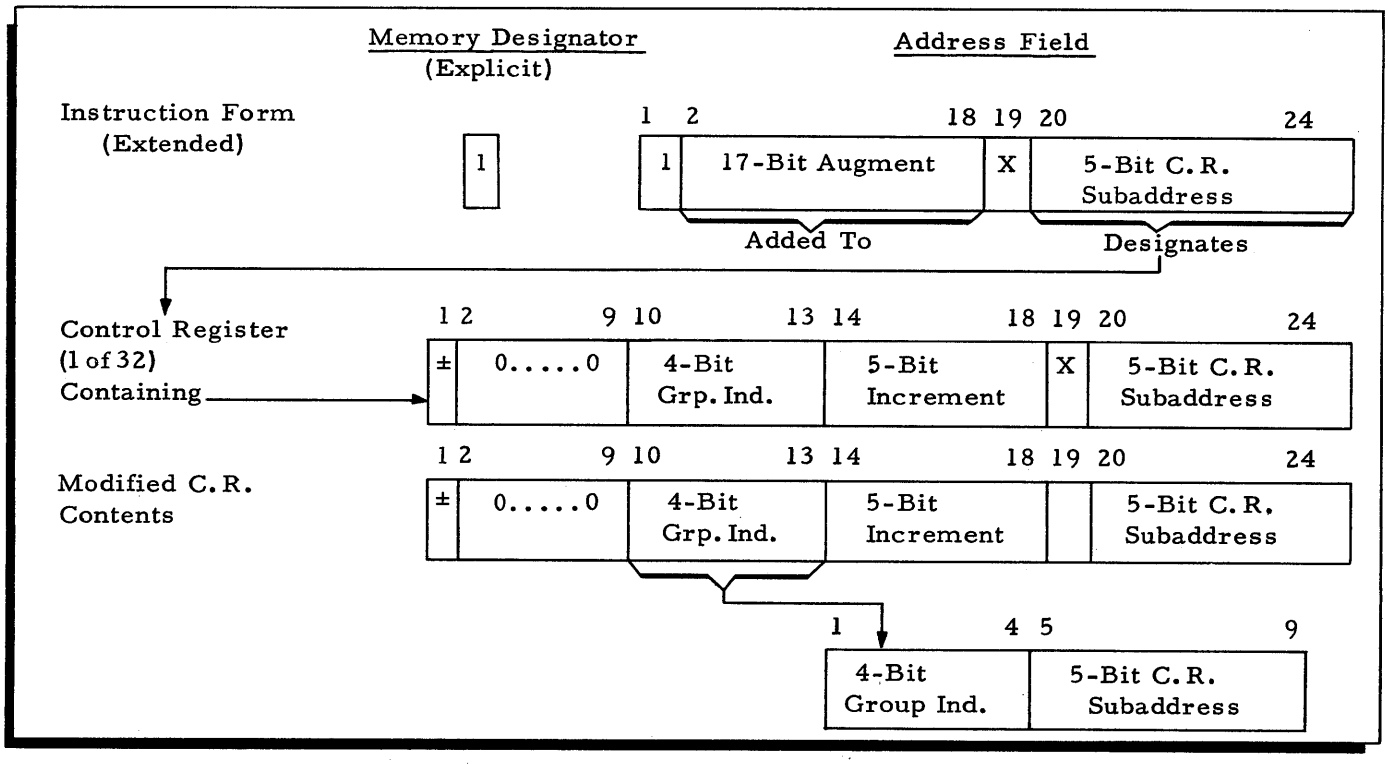


Figure 5-18. Extended Indexed Control Register Addressing

Indirect Memory Location Address (Normal and Extended)

In some instances, it is useful to be able to specify in the address field of the instruction the address of a control register where the main memory address of the desired operand or result is stored, rather than to specify the location of either directly. This method of locating an operand or result is called indirect memory location addressing.

In the normal mode of addressing, the bit configuration of the address field is identical to that described under control register addressing with the exception that the tabular bit (bit 7) in the address field has the value of one rather than zero. Since the memory designator bit must also have the value of one (control memory), this type of addressing may be used only in the unmasked versions of the arithmetic, logical, decision, and information transfer instructions, the inherent-mask instructions, and the scientific instructions. The control register whose address is generated, as shown in Figure 5-19, does not contain the operand or the result location for the instruction, but the main memory address to be used.

The address generated is that of a control register in the same group as the sequencing counter which selected the instruction. The contents of this control register are interpreted as a 23-bit (signed) main memory address. Thus, indirect memory location addressing, like indexed memory location addressing, provides access to operands and results in any bank of memory regardless of the bank indicator stored in the sequencing counter. After the contents

of the control register have been used to locate the desired operand or result, the stored contents are permanently modified by the increment specified in the address field of the instruction, under direction of the control register sign. This incrementing (or decrementing) takes place regardless of whether the memory location addressed is an operand or a result location.

Indirect addressing is a useful tool for stepping sequentially through an array of items, processing the n^{th} word of each item. The location of word n of the first item is stored in a control register. When this location is addressed indirectly, the use of the proper increment (equal to the item size) sets the contents of the control register to the location of word n of the second item, and so forth, until word n of each item has been processed.

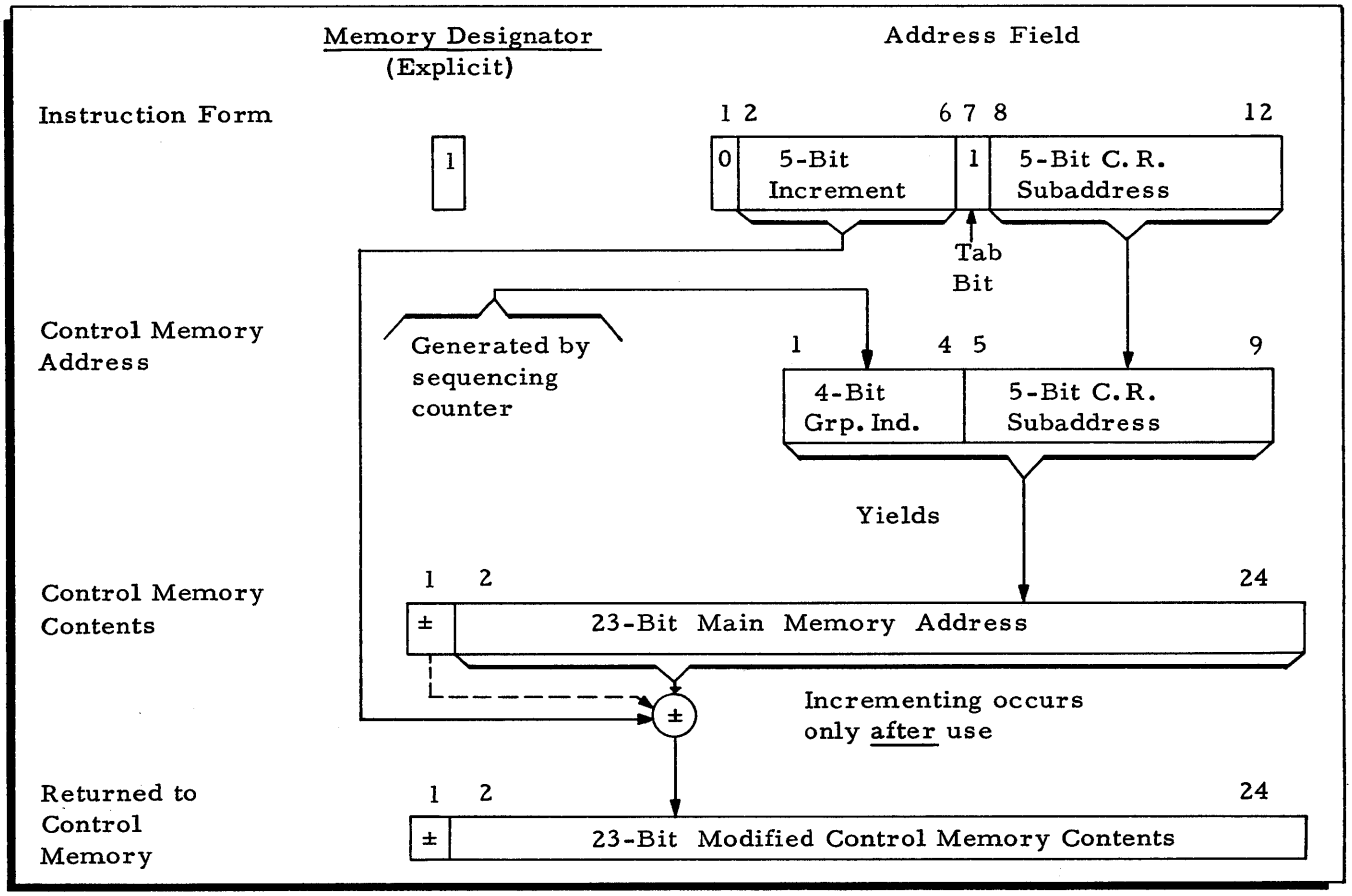


Figure 5-19. Normal Indirect Memory Location Addressing

In the extended mode, if the high-order bit of the address field is a zero, and both the corresponding memory designator bit and the tabular bit are ones, then the contents of the control register, as specified by the five low-order bits of the address field and the controlling group, form the extended main memory address (see Figure 5-20).

After use, the contents of the specified register are incremented by the value of bits 2 through 18 of the address field.

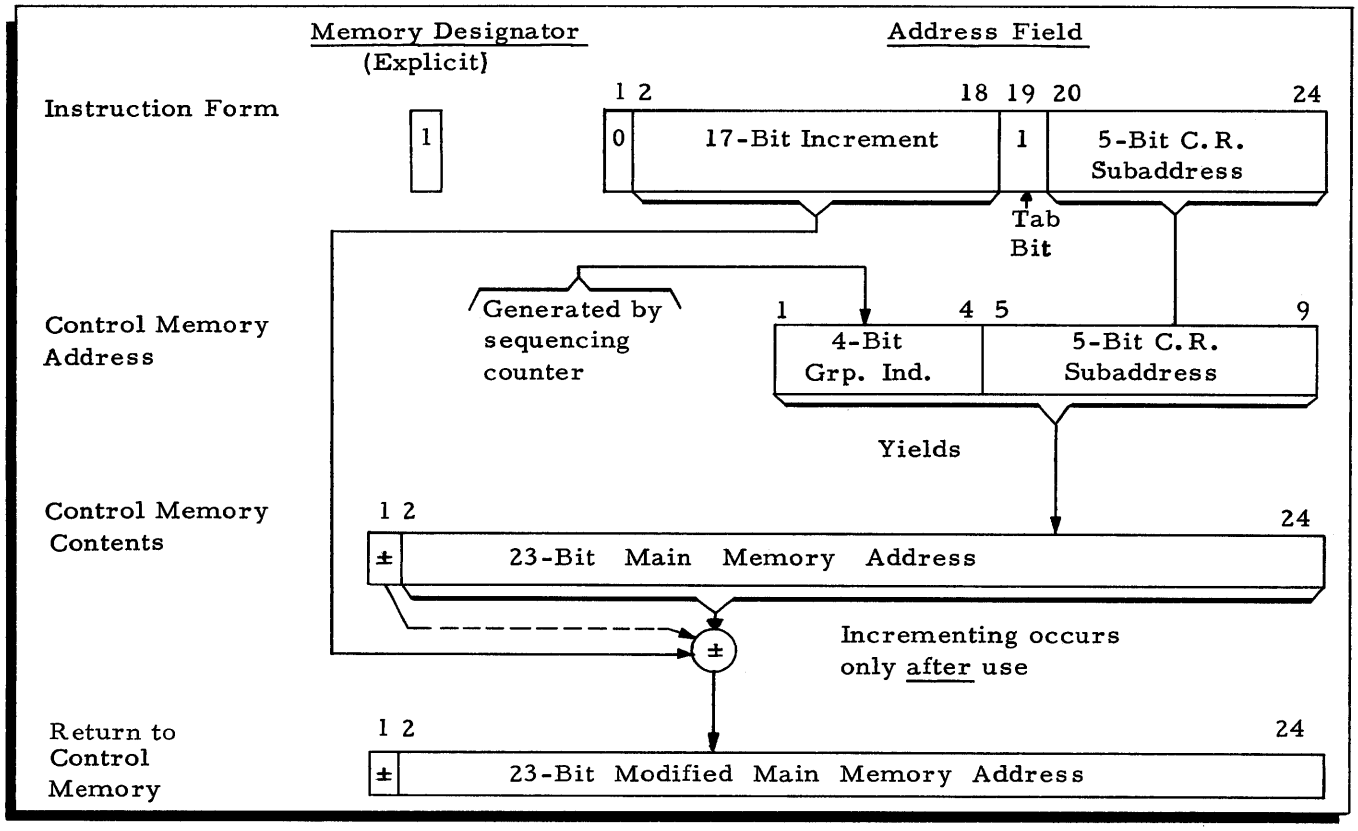


Figure 5-20. Extended Indirect Memory Location Addressing

Mod 8 assembly language recognizes the use of indirect memory location addressing by the notation

N, Control Register Designator, Increment

where N represents a memory designator bit of one in the command code, a zero in the first bit position of the address field, and a one in the tabular bit position of the address field. The control register designator specifies one of the registers in the related group, either numerically or mnemonically (see Table 5-2), and the increment is a number from 0 to 31 (0 to 131,071 in extended mode). The computer interprets the contents of the specified register as the bank indicator and subaddress of a memory location in any bank. The increment is added to the low-order bits of the contents of the control register after use, permanently altering them. Thus, the address

N, R1, 10

designates the contents of the related control register R1, which are interpreted as the location of an operand in main memory. After use, the contents of R1 are incremented by 10.

Indexed Indirect Memory Location Address (Normal and Extended)

In the normal mode, if the high-order bit of this address field is a one and the corresponding memory designator is a one, then the address field is interpreted as an 8-bit augment and a 3-bit index register number. Figure 5-21 illustrates the generation of an indexed indirect memory location address in the normal mode. The contents of the index register specified (which is taken from the same group from which the current instruction was selected) is augmented, under control of its sign, by the low-order eight bits of the address field. The contents of the modified index register are interpreted as a 5-bit control register subaddress, a tabular bit, a 5-bit increment, and a 3-bit group indicator, plus a sign. The remaining nine bits are considered to be zeros. The 5-bit subaddress and the 3-bit group indicator, together with a high-order zero are combined to form a complete 9-bit control memory address. If the tabular bit was a one, the contents of the specified control memory location are interpreted as a 24-bit main memory address designating an operand location or a result destination.

Indexed indirect memory location addressing makes it possible to use any of the control registers in a controlling group to address any available memory location indirectly. The retained contents of the control register are always modified after use by the amount of the increment, under control of the control register sign bit.

Extended indexed indirect addressing proceeds exactly as noted in the discussion for extended control register addressing (refer to Figure 5-18 for the illustration of the address generated) except, when the tabular bit of the augmented control register is a one, then the contents of the control register specified subsequently are used to locate an operand in main memory. As in the case of extended indexed control register addressing, bit 1 is the sign bit, the control memory operands contain 23 bits, and the group number is always 4 bits (bits 10 through 13).

The assembly language notation for an indexed indirect location address (see page 5-26) resembles that for an indexed control register address, taking the form

Index Register Designator, N, Control Register Designator, Increment.

The comments made with respect to assembly notation for an indexed control register address are also applicable to an assembly indexed indirect memory location address.

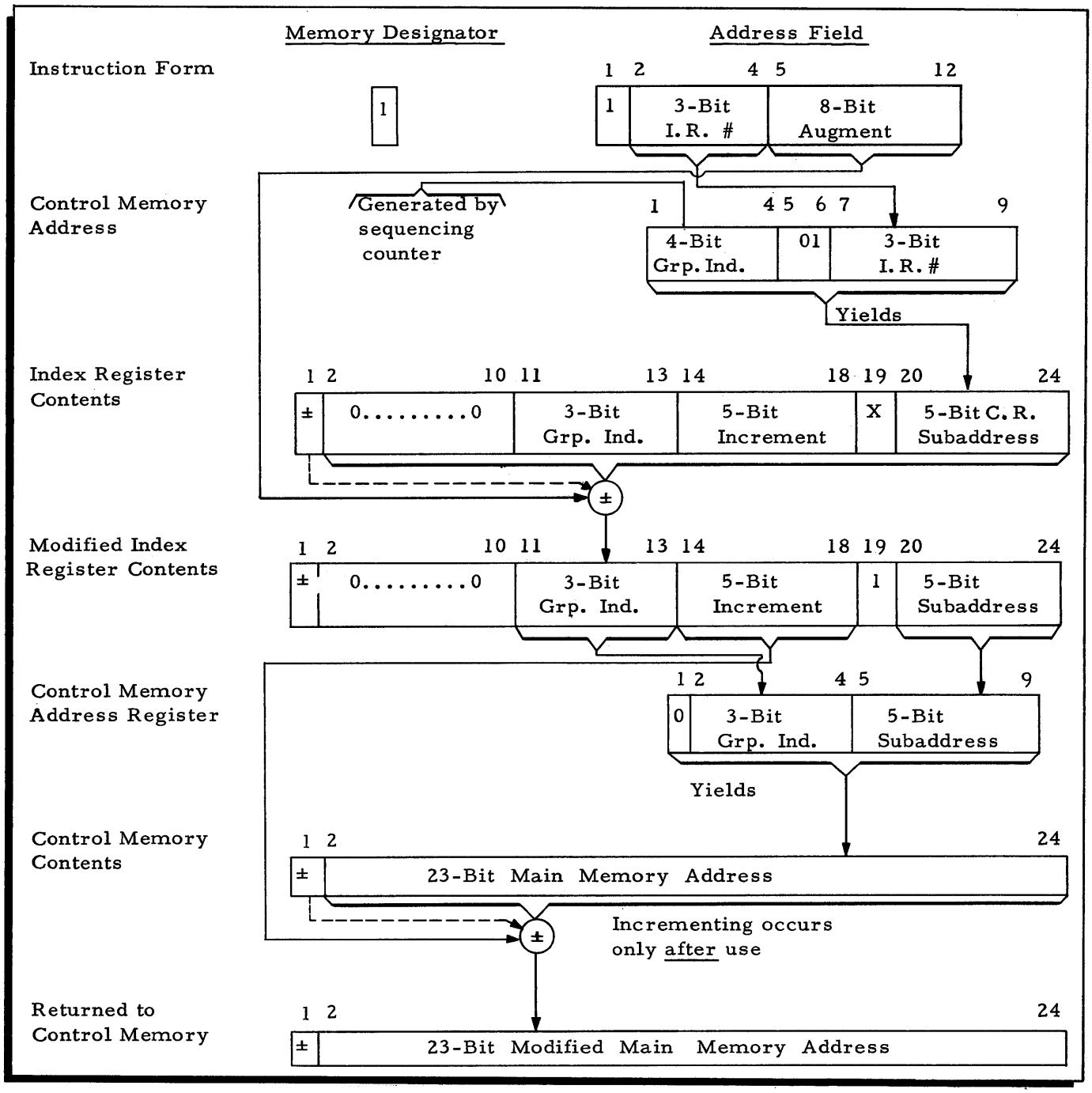


Figure 5-21. Normal Indexed Indirect Memory Location Addressing

Summary of Address Forms

The binary forms of six different address types were described in the preceding pages and illustrated in Figures 5-7 through 5-21. Figure 5-22 suggests a method with which the reader may determine by inspection the address type of any binary address configuration. This figure is not illustrative of the steps taken by the machine in interpreting addresses.

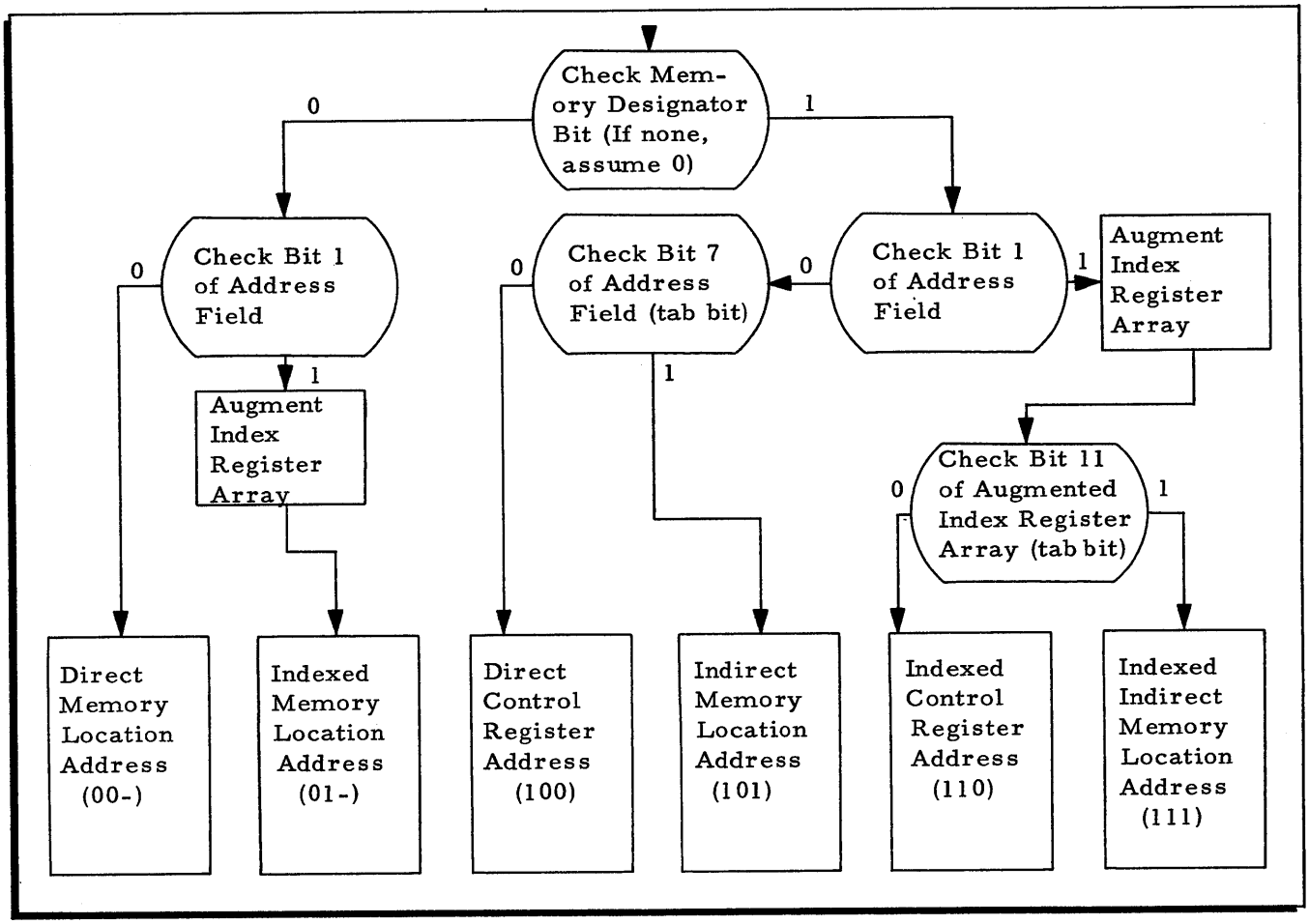


Figure 5-22. Interpretation of Address Bit Structure

Inactive Addresses

The word processor contains three arithmetic registers which have no addresses: the accumulator, the mask register, and the low-order product register. Programmer access to these registers is provided by the technique of inactive addressing with certain specified instructions. When an address field in an instruction contains all binary ones, that address is said to be inactive. The memory designator bit, if any, must be zero; otherwise, the behavior of the system is unspecified. Instructions in which the designator bits are used for other purposes and do not have to be zero are: optional-mask instructions, input/output instructions, and simulator instructions. In addition, the designator bit for the A-address field of the Control Program instruction can be either one or zero, because the A-address field is ignored in this instruction.

Access to the accumulator is provided by the proper use of inactive addressing in conjunction with the add instructions. Inactive addressing with the Extract instruction provides access to the mask register. Access to the low-order product register is made possible by inactive addressing with the Transfer and Sequence Change instruction (TS in assembly language).

The behavior of the accumulator when inactive addressing is used in the Binary Add, Decimal Add, or Word Add instruction is specified as follows:

1. If address A is inactive, the previous contents of the accumulator are used as if they were the contents of A.
2. If address B is inactive, the accumulator (which contains the contents of A if A is active or the previous contents of the accumulator if A is inactive) is left undisturbed.
3. If address C is inactive, the normal process of hunting for the next sequence counter in demand is inhibited, and the result remains in the accumulator at the conclusion of the instruction.

Thus, if the B and C addresses are inactive, the effect of the instruction is to transfer the contents of address A to the accumulator. If the A and B addresses are inactive, on the other hand, the effect is to transfer the contents of the accumulator to the location specified in the C address. Certain restrictions should be noted with respect to the sequencing of these instructions when they contain inactive addresses. If a Decimal or Binary Add instruction is used to place a word in the accumulator, the same type of instruction should be used to transfer the contents of the accumulator, with proper sign, to memory. Similarly, if a word has been placed in the accumulator by a Word Add instruction, the Word Add instruction must be used to deliver the contents of the accumulator to memory. The explanation for this restriction is reserved for the section discussing the arithmetic instructions in detail.

The mask register, not to be confused with the mask index registers in the control memory, is a full-word register in the word processor which stores the mask during the execution of a masked instruction. In the Extract instruction, the location of the mask is specified in the B address. At the conclusion of the execution of an Extract instruction with all addresses active, the original contents of the B address are left in the mask register. Thus, a word may be loaded into the mask register, without disturbing memory, by using an Extract instruction with an inactive C address. If address B is inactive, the previous contents of the mask register will be used as the mask. The contents of the mask register are transferred to the location specified in address C of an Extract instruction if address B is inactive and the contents of address A consist of all (48) binary ones.

The low-order product register is of interest to the programmer primarily because it stores the low-order portion of the result of a multiply instruction. In order to obtain this result, the programmer may use the Transfer and Sequence Change instruction (TS) with an inactive A address, which transfers the contents of the low-order product register to the location specified by the B address. If address A of this instruction is active and address B is inactive, the contents of A are transferred to the low-order product register and to the accumulator.

If any instructions intervene between the instruction which stores information in an arithmetic register and that which attempts to retrieve the information, there is no guarantee that the contents will remain. Moreover, if a masked instruction is used to attempt to retrieve information from the accumulator or low-order product register, the behavior of the system is unspecified unless the same mask was used when the information was stored there. Since masking is not permitted with the multiply instructions, a masked Transfer and Sequence Change (TS) instruction should never be used to retrieve the low-order product.

Two other general situations in which the effect of inactive addressing has been specified should be mentioned:

1. If the C-address field is inactive in an instruction that normally changes the sequencing counter to select the next instruction from the location given in the C-address field, the counter remains unchanged, unless its contents have been altered under the direction of the A- or B-address field.
2. When an inactive C address occurs in any instruction for which such an address is allowed, the normal process of hunting for the next sequencing counter in demand is omitted.

For further information on inactive addressing not specifically mentioned in this discussion, refer to the tables of inactive addressing in each instruction grouping.

In the assembly language, an inactive address is indicated by a hyphen (-) in the address field.

INSTRUCTIONS

The remainder of this section describes the instructions used by the word processor. The symbols used in defining these instructions are listed below.

FORMAT

B	Bisequence bit. If this bit is a 0, the next instruction will be taken from the sequence counter; if it is a 1, the next instruction will be taken from the cosequence counter.
a	Memory designator bit for A address. If bit value is 0, address refers to main memory. If bit value is 1, address refers to control memory.
b	Memory designator bit for B address. If bit value is 0, address refers to main memory. If bit value is 1, address refers to control memory.
c	Memory designator bit for C address. If bit value is 0, address refers to main memory. If bit value is 1, address refers to control memory.
x	Irrelevant.

PPPPPP	Peripheral address.
y	$0\text{-}3_{(8)}$
z	$4\text{-}7_{(8)}$
$m^1 \ldots m^6$	Shift mask subaddress bit positions.
$p^1 \ldots p^6$	Parameter bits.

MASKING

MM MMM	Field mask subaddress.

LEGAL ADDRESS TYPES

p	The field must contain one or more parameters, such as a number indicating the quantity of words to be transferred. The bit positions are indicated by the superscripts, e.g., $p^1 \ldots p^6$.
D	Direct memory location addressing may be used in this field.
I	Indexed memory location addressing may be used in this field.
H	Indirect memory location addressing may be used in this field.
IH	Indexed indirect memory location addressing may be used in this field.
All	Includes D, I, H, and IH, plus direct control register, indexed control register, and extended address types.

ARITHMETIC INSTRUCTIONS

Arithmetic instructions in the word processor involve the use of two arithmetic registers called the accumulator (AC) and the low-order product register (LOP). These registers are accessible to the programmer through the technique of inactive addressing with certain specified instructions. The accumulator is used in all the instructions described below. The low-order product register is used in the multiply instructions and in the masked arithmetic instructions.

The Accumulator

The accumulator consists of a 48-bit register, capable of storing an entire word. Used in conjunction with the accumulator is a single bit called the sign position. Since the operands handled in most arithmetic operations are treated as 44-bit numbers with 4-bit signs, most arithmetic instructions use the sign position together with the low-order 44 bits of the accumulator. The exceptional instructions which handle unsigned 48-bit numbers are so noted as they are described.

For all arithmetic operations on signed numbers, the sign position, originally set to the value of zero, is changed to the value of one if any of the four sign bits in the A operand is a one. (This is the logical OR function.) The setting of the sign position, the sign of the B

operand, and the operation code determine the sign of the result. When the result is read out of the accumulator, a sign consisting of four bits identical in value (zero or one) to the final setting of the sign position is attached to the low-order 44 bits of the accumulator. Thus, only two sign configurations may be obtained as the result of an arithmetic operation: four binary zeros indicating a negative result or four binary ones indicating a positive result. If the result of an add or subtract instruction is zero, the result takes a sign based on the sign of the A operand.

When addition is performed on signed numbers, bits 1 through 4 of the accumulator are automatically filled with binary ones so that if overflow occurs it may be sensed in bit 1, the same position in which it is sensed during the addition of unsigned 48-bit operands. Similarly, bits 1 through 4 are filled with binary zeros when subtracting signed numbers in order that borrows may be sensed at the same position for both signed and unsigned numbers. If overflow occurs, the instruction is completed and the low-order 44 bits of the accumulator, plus a 4-bit sign based on the value of the sign position, are stored in the location specified by the C address. An unprogrammed transfer is then made to U + 8 or U + 9, where the programmer should have stored the entry to a subroutine to handle this condition. The instruction which resulted in the overflow is stored in U or U + 1 (see Table 5-3).

The arithmetic operations are not actually performed in the accumulator but in a 48-bit parallel adder. Both binary and decimal arithmetic are performed in the same adder, which handles twelve 4-bit groups at a time. For binary instructions, these 4-bit groups are considered as hexadecimal digits, with carry occurring after a group reaches the value of 15. This results in a pure binary operation. For decimal instructions, the adder is made to carry when a 4-bit group reaches the value of nine, so that decimal arithmetic is performed. If a Decimal Addition instruction involves an operand which contains hexadecimal digits, however, a variant on normal addition occurs in accordance with the following rules:

1. If the hexadecimal digit appears in the A operand, the corresponding digit in the B operand is added in hexadecimal fashion. In other words, 14 + 1 becomes 15, 14 + 3 becomes 1 with a carry of 1.
2. If the A operand contains decimal information and a hexadecimal digit occurs in the B operand, then the result is unspecified and a mod-3 error may occur.[1]

The contents of the operands are inspected digit by digit. Therefore, the result obtained by adding two words having both hexadecimal and decimal digits must be ascertained on a digit-by-digit basis. Since this condition is not considered an error by the word processor, except in the circumstance noted above, the programmer will receive no indication of the existence of a hexadecimal digit in an operand handled by a decimal instruction.

[1] For a description of mod-3 checking, refer to page B-1.

Several precautions must be observed in working with the accumulator. In the discussion of inactive addressing, it is pointed out that the result of an addition may be left in the accumulator by using an add instruction with an inactive C address and that the contents of the accumulator may be stored in memory by using an add instruction with inactive A and B addresses. It should also be noted that if the contents of the accumulator were formed by an instruction which treated the operands as signed 44-bit numbers, such an instruction must be used to store the contents of the accumulator in order to guarantee them the proper sign. Otherwise, the entire 48-bit contents will be stored rather than the low-order 44 bits with a 4-bit sign determined from the value of the sign position. Similarly, if the contents of the accumulator were created by an instruction which treated the operands as unsigned 48-bit words, such an instruction must be used to transfer the entire contents of the accumulator to memory without reference to the sign position.

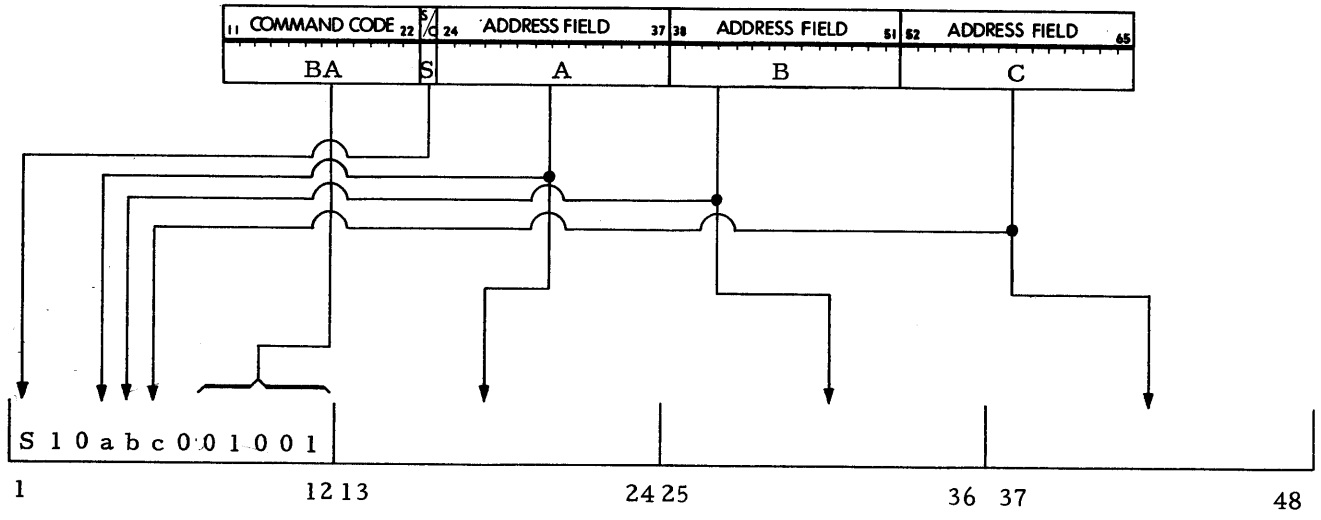
Another precaution involves the condition of the accumulator after its contents have been delivered to memory. After a result formed in the accumulator has been transferred to memory, there is no guarantee that the contents of the accumulator will remain, since a hunt for the next program demand will have occurred immediately after the first transfer.

The Low-Order Product Register

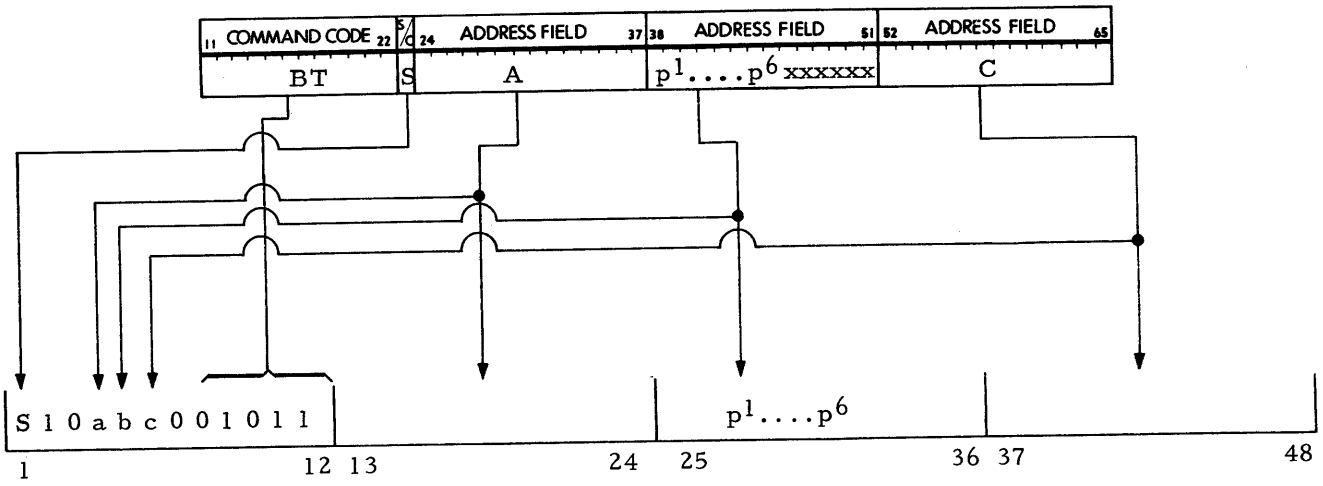
The low-order product register is a 48-bit register similar to the accumulator. As its name implies, the register is used to store the low-order portion of the result of a multiply instruction. The contents of the register are interpreted as a sign and a 44-bit number.

Assembly

With the exception of the Binary Accumulate (BT) and the Decimal Accumulate (DT) instructions, all the instructions in the arithmetic grouping are assembled as indicated in the following diagram (using the Binary Add instruction):



The following diagram (using the Binary Accumulate instruction) displays the instruction assembly for the Binary Accumulate and Decimal Accumulate. Variation from the above example occurs in the B address field, which in these instances, contains parameters.



BA	BINARY ADD

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B10 abc 001 001 $(2011)_8$	A	B	C

FUNCTION

The Binary Add instruction causes the contents of the location specified by A to be added algebraically to the contents of the location specified by B and the result of the operation to be stored in the location specified by C. The contents of both A and B are regarded as 44-bit numbers with 4-bit signs. If any sign bit for the A or B operand is a one, then the corresponding operand is considered positive. After the addition is complete, the low-order 44 bits of the accumulator, plus a 4-bit sign (1111 or 0000) corresponding to the value (1 or 0) of the sign position, are stored in the location specified by C. If overflow occurs, the instruction is stored in U if the sequence counter selected it or in U + 1 if the cosequence counter selected it, and the next instruction is taken from U + 8 or U + 9.

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 101 001 $(0051)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D, I	D, I	D, I
Unmasked	All	All	All

Inactive:

Both masked and unmasked versions of the Binary Add instruction may use inactive addressing in the manner described in Tables 5-4 and 5-5 on pages 5-51 and 5-52.

EBA | EXTENDED BINARY ADD

FORMAT

1	12	13	24	25	36	37	48
Command Code		Address Field		Address Field		Address Field	
B11 abc 001 011 $(3013)_8$		A		B		C	

FUNCTION

The Extended Binary Add instruction is identical to the Binary Add instruction unless an address field contains a control memory address. In this case, all 24 bits of the relevant control memory register are used in the operation, regardless of the mode of the instruction.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
	All	All	All

Inactive:

The Extended Binary Add instruction may use inactive addressing in the manner described in Table 5-4, on page 5-51.

BS | BINARY SUBTRACT

FORMAT

1	12	13	24	25	36	37	48
Command Code		Address Field		Address Field		Address Field	
B10 abc 011 001 $(2031)_8$		A		B		C	

FUNCTION

The Binary Subtract instruction causes the contents of B to be subtracted algebraically from the contents of A and the results to be stored in C. The contents of both A and B are regarded as 44-bit numbers with 4-bit signs. If any sign bit for the A or B operand is a one, then the corresponding operand is considered positive. After the subtraction is complete, the low-order 44 bits of the accumulator, plus a 4-bit sign (1111 or 0000) corresponding to the value (1 or 0) of the sign position, are stored in the location specified by C. If overflow occurs, the instruction is stored in U if the sequence counter selected it or in U + 1 if the cosequence counter selected it, and the next instruction is taken from U + 8 or U + 9.

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 111 001 $(0071)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D, I	D, I	D, I
Unmasked	All	All	All

Inactive:

Both masked and unmasked versions of the Binary Subtract instruction may use inactive addressing in the manner described in Tables 5-4 and 5-5, on pages 5- 51and 5-52.

EBS | EXTENDED BINARY SUBTRACT

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B11 abc 011 011 $(3033)_8$	A	B	C

FUNCTION

The Extended Binary Subtract instruction is identical to the Binary Subtract instruction unless an address field contains a control memory address. In this case, all 24 bits of the relevant control memory register are used in the operation, regardless of the mode of instruction.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
	All	All	All

Inactive:

The Extended Binary Subtract instruction may use inactive addressing in the manner described in Table 5-4, on page 5-51.

DA | DECIMAL ADD

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B10 abc 000 001 $(2001)_8$	A	B	C

FUNCTION

The Decimal Add instruction causes the contents of A to be added algebraically to the contents of B and the result of the operation to be stored in C. The instruction differs from the Binary Add instruction only in the fact that the contents of each operand are handled as eleven 4-bit groups with a 4-bit sign. If any sign bit for the A or B operand is a one, then the corresponding operand is considered positive. After the addition is complete, the low-order 44 bits of the accumulator, plus a 4-bit sign (1111 or 0000) corresponding to the value (1 or 0) of the sign position, are stored in the location specified by C. If overflow occurs, the instruction is stored in U if the sequence counter selected it, or in U + 1 if the cosequence counter selected it, and the next instruction is taken from U + 8 or U + 9.

NOTE: If the A operand contains decimal information and a hexadecimal digit occurs in the B operand, then the result is unspecified and a mod-3 error may occur. (Refer to page B-1 for a description of mod-3 checking.)

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 100 001 $(0041)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D,I	D,I	D,I
Unmasked	All	All	D, I, H, IH

Inactive:

Both masked and unmasked versions of the Decimal Add instruction may use inactive addressing in the manner described in Tables 5-4 and 5-5, on pages 5-51 and 5-52.

DS | DECIMAL SUBTRACT

FORMAT

1	12	13	24	25	36	37	48
Command Code		Address Field		Address Field		Address Field	
B10 abc 010 001 $(2021)_8$		A		B		C	

FUNCTION

The Decimal Subtract instruction causes the contents of B to be subtracted algebraically from the contents of A and the results to be stored in C. The instruction differs from the Binary Subtract instruction only in the fact that the contents of each operand are handled as eleven 4-bit groups with a 4-bit sign. If any sign bit for the A or B operand is a one, then the corresponding operand is considered positive. After the subtraction is complete, the low-order 44 bits of the accumulator, plus a 4-bit sign (1111 or 0000) corresponding to the value (1 or 0) of the sign position, are stored in the location specified by C. If overflow occurs, the instruction is stored in U if the sequence counter selected it or in U + 1 if the cosequence selected it, and the next instruction is taken from U + 8 or U + 9.

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 110 001 $(0061)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D,I	D,I	D,I
Unmasked	All	All	All

Inactive:

Both masked and unmasked versions of the Decimal Subtract instruction may use inactive addressing in the manner described in Tables 5-4 and 5-5, on pages 5-51 and 5-52.

WA | WORD ADD

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B10 abc 001 101 $(2015)_8$	A	B	C

FUNCTION

Word Add is an arithmetic instruction which regards operands as unsigned 48-bit numbers. The instruction adds the absolute values of the entire 48-bit content of A and B in binary and stores the entire 48-bits of the accumulator in C, making no reference to the sign position. Overflow is sensed in bit 1. If overflow occurs, the instruction is completed, stored in U if the sequence counter selected it or in U + 1 if the cosequence counter selected it, and the next instruction is taken from U + 8 or U + 9.

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 101 101 $(0055)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D,I	D,I	D,I
Unmasked	All	All	All

Inactive:

Both masked and unmasked versions of the Word Add instruction may use inactive addressing in the manner described in Tables 5-4 and 5-5, on pages 5-51 and 5-52.

WD | WORD DIFFERENCE

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B10 abc 011 101 $(2035)_8$	A	B	C

FUNCTION

Word Difference is an arithmetic instruction which treats its operands as unsigned 48-bit numbers. The instruction causes the entire 48-bit contents of B to be subtracted in binary from the entire 48-bit contents of A. The entire 48 bits of the accumulator are stored in C, making no reference to the sign position. If the absolute value of the contents of B is greater than the absolute value of the contents of A, then overflow occurs, and the result stored in C is the difference of the absolute values of the words.

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 111 101 $(0075)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D,I	D,I	D,I
Unmasked	All	All	All

Inactive:

Both masked and unmasked versions of the Word Difference instruction may use inactive addressing in the manner described in Tables 5-4 and 5-5, on pages 5-51 and 5-52.

BT	BINARY ACCUMULATE

FORMAT

1 12	13 24	25 36	37 48
Command Code	Address Field	Address Field	Address Field
B10 abc 001 011 $(2013)_8$	A	$p^1 \ldots p^6$ xxxxxx	C

FUNCTION

The Binary Accumulate instruction totals the absolute value of the contents of A the number of times specified by the high-order six bits of the B-address field, a number which ranges from 0 through 63. Although the words added are treated as signed 44-bit numbers, only their absolute values are added. The accumulator is not cleared between successive additions except for the high-order four bits. At the conclusion of the series of additions, the 44 low-order bits of the accumulator are stored in C, together with a sign (four binary ones or four binary zeros) based on the value of the sign position. This value represents the sign of the first word added (the contents of the location originally specified by the A address field).

Step by step, the instruction functions as follows. The low-order 44 bits of the A operand are transferred to the accumulator and the high-order four bits of the accumulator are set to one. If A contains a control register subaddress, incrementing is performed as specified. The high-order four bits of the accumulator are again replaced with ones, and the low-order 44 bits of A are added in binary to the contents of the accumulator. (Note that the location now specified by A will be different from the original A if incrementing took place.) The specified incrementing is again performed (if A contains a control register subaddress), the high-order four bits of the accumulator are replaced by ones and the low-order 44 bits of A are again added to the accumulator. The process is performed the number of times specified by the high-order six bits of the B-address field. If the value of these bits is zero, no information is transferred to the accumulator, the instruction is not executed, and the next instruction is selected from the sequencing counter specified by bit 1 of the command code. The low-order six bits of the B-address field are ignored.

Overflow in the accumulator is sensed in bit 1. If overflow is sensed, the instruction is completed and stored in U if the sequence counter selected it or in U + 1 if the cosequence counter selected it, and the next instruction is taken from U + 8 or U + 9. Since the high-order four bits of the accumulator are replaced by ones between successive additions, the contents of these four positions are not available to indicate the number of overflows. Thus, unless the programmer knows from the logic of his problem precisely how many overflows may have occurred and at which points, he must repeat the addition process in pairs.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	p	All

Inactive:

The Binary Accumulate instruction may use inactive addressing in the manner described in Table 5-6, on page 5-53.

DT	DECIMAL ACCUMULATE

FORMAT

1 – 12	13 – 24	25 – 36	37 – 48
Command Code	Address Field	Address Field	Address Field
B10 abc 000 011 $(2003)_8$	A	$p^1 \ldots p^6$ xxxxxx	C

FUNCTION

The Decimal Accumulate instruction totals the absolute value of the contents of A the number of times specified by the high-order six bits of the B-address field, a number which ranges from 0 through 63. The contents of each operand are handled as eleven 4-bit groups with a 4-bit sign and they are added according to the rules of decimal arithmetic. The accumulator is not cleared between successive additions except for the high-order four bits. If hexadecimal digits appear in the operands, they are added in the same fashion described under "The Accumulator" on page 5-36. At the conclusion of the successive additions, the low-order 44 bits of the accumulator, plus a 4-bit sign (1111 or 0000) corresponding to the value (1 or 0) of the sign position, are stored in the location specified by C. This value represents the sign of the first word added (the contents of the location originally specified by the A-address field).

Step by step, the instruction functions as follows. The low-order 44 bits of the A operand are transferred to the accumulator and the high-order four bits of the accumulator are set to one. If A contains a control register subaddress, incrementing is performed as specified. The high-order four bits of the accumulator are again replaced with ones, and the low-order 44 bits of the A are added in decimal to the contents of the accumulator. (Note that the location now specified by A will be different from the original A if incrementing took place.) The specified incrementing is again performed (if A contains a control register subaddress), the high-order four bits of the accumulator are replaced by ones, and the low-order 44 bits of A are again added to the accumulator. This process is performed the number of times specified by the high-order six bits of the B-address field. If the value of these bits is zero, no information is transferred to the accumulator, the instruction is not executed, and the next instruction is selected from the sequencing counter specified by bit 1 of the command code. The low-order six bits of the B-address field are ignored.

Overflow in the accumulator is sensed in bit 1. If overflow is sensed, the instruction is completed and stored in U if the sequence counter selected it or in U + 1 if the cosequence counter selected it, and the next instruction is taken from U + 8 or U + 9. Since the high-order four bits of the accumulator are replaced by ones between successive additions, the contents of these four positions are not available to indicate the number of overflows. Thus, unless the programmer knows from the logic of his problem precisely how many overflows may have occurred and at which points, he must repeat the addition process in pairs.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	p	All

Inactive:

The Decimal Accumulate instruction may use inactive addressing in the manner described in Table 5-6, on page 5-53.

BM	BINARY MULTIPLY

FORMAT

1	12	13	24	25	36	37	48
Command Code		Address Field		Address Field		Address Field	
B00 abc 001 011 $(0013)_8$		A		B		C	

FUNCTION

The Binary Multiply instruction multiplies the absolute value of the contents of the location specified in A by the contents of the location specified in B. The contents of both A and B are regarded as 44-bit numbers with 4-bit signs. After multiplication is completed, the high-order 44-bit product is stored with the proper sign (0000 or 1111), based on the value of the sign position, in both C and the accumulator. The low-order 44-bit product is stored with the proper sign (0000 or 1111), based on the value of the sign position, in the low-order product register.

Since hunting for the next sequencing counter in demand is not allowed at the conclusion of a Multiply instruction, the programmer has the opportunity to store the contents of the low-order product register before an instruction from another program destroys them; this may be accomplished by performing a Transfer and Sequence Change (TS) instruction with an inactive A address. If both halves of the product are stored, they will have the same sign. It should be noted that the high-order product stored in C is unrounded. If the C address of a Binary Multiply instruction is a direct or indexed control register address, an error will result and a call to master control will occur. It should also be noted that a Binary Multiply instruction with an inactive C address will store the contents of A in location 0, (program address) with bad parity.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	D, I, H, IH

Inactive:

The Binary Multiply instruction may use inactive addressing in the manner described in Table 5-7, on page 5-53.

DM	DECIMAL MULTIPLY

FORMAT

1	12	13	24	25	36	37	48
Command Code		Address Field		Address Field		Address Field	
B00 abc 000 011 $(0003)_8$		A		B		C	

FUNCTION

The Decimal Multiply instruction multiplies the absolute value of the contents of the location specified in A by the contents of the location specified in B. The contents of each operand are handled as eleven 4-bit groups with a 4-bit sign. Upon completion, the Decimal Multiply instruction produces a 2-word result. The high-order product, consisting of 11 decimal digits, is stored in the location specified by the C-address field with a sign (0000 or 1111) determined by the value of the sign position. The low-order result is stored with the proper sign (0000 or 1111), based on the value of the sign position, in the low-order product register. Since hunting for the next sequencing counter in demand is not allowed at the conclusion of a multiply instruction, the programmer has the opportunity to store the contents of the low-order product register before an instruction from another program destroys them; this may be accomplished by performing a Transfer and Sequence Change (TS) instruction with an inactive A address. If both halves of the product are stored, they will have the same sign. It should be noted that the high-order product stored in C is unrounded. If hexadecimal digits appear in the operands, an erroneous product will be generated, and an error may be indicated. Note that if the C address of a Decimal Multiply instruction is a direct or indexed control register address, an error will result and a call to master control will occur.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	D, I, H, IH

Inactive:

The Decimal Multiply instruction may use inactive addressing in the manner described in Table 5-7, on page 5-53.

BD | BINARY DIVIDE

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B00 abc 000 101 $(0005)_8$	A	B	C

FUNCTION

The Binary Divide instruction performs a binary division of the contents of B (dividend) by the contents of A (divisor), delivering the quotient to the location specified by C; the system hunts for the next sequencing counter in demand. The quotient is also retained in the accumulator and the remainder in the low-order product register.

If the absolute value of the contents of B is equal to or greater than the absolute value of the contents of A, the instruction is not performed but is stored in U or U + 1. Division overcapacity unprogrammed transfer is executed to U + 10 or U + 11. The behavior of the system is unspecified if the program attempts to retrieve a result following division overcapacity by means of a Multiple Unload instruction.

If C is inactive, no result is delivered to main memory (the high-order and low-order parts are retained in the accumulator and the low-order product register, respectively), and hunting is inhibited.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:

	A	B	C
Unmasked	All	All	All

Inactive:

The Binary Divide instruction may use inactive addressing in the manner described in Table 5-8, on page 5-54.

DD | DECIMAL DIVIDE

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B00 abc 010 101 $(0025)_8$	A	B	C

FUNCTION

The Decimal Divide instruction performs a decimal division of the contents of B (dividend) by the contents of A (divisor), delivering the quotient to the location specified by C; the system hunts for the next sequencing counter in demand. The quotient is also retained in the accumulator and the remainder in the low-order product register.

If the absolute value of the contents of B is equal to or greater than the absolute value of the contents of A, the instruction is not performed but is stored in U or U + 1. Division overcapacity unprogrammed transfer is executed to U + 10 or U + 11. The behavior of the system is unspecified if the program attempts to retrieve a result following division overcapacity by means of a Multiple Unload instruction.

If C is inactive, no result is delivered to main memory (the high-order and low-order parts are retained in the accumulator and the low-order product register, respectively, and hunting is inhibited.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Decimal Divide instruction may use inactive addressing in the manner described in Table 5-8, on page 5-54.

Table 5-4. Unmasked Inactive Addressing for Add and Subtract Instructions

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	No operation takes place. [1]
INACTIVE	INACTIVE	ACTIVE	The contents of the accumulator are delivered to the location specified by C with the sign bits (0000 or 1111), depending upon the value of the sign position.
INACTIVE	ACTIVE	INACTIVE	The contents of the accumulator are added to or subtracted from the contents of B and placed in the accumulator. [1]
INACTIVE	ACTIVE	ACTIVE	The contents of the accumulator are added to or subtracted from the contents of the B and delivered to the location specified by C.
ACTIVE	INACTIVE	INACTIVE	The contents of the location specified by A are placed in the accumulator. [1]
ACTIVE	INACTIVE	ACTIVE	The contents of the location specified by A are delivered to the location specified by C.

Table 5-4 (cont). Unmasked Inactive Addressing for Add and Subtract Instructions

A	B	C	Action for void addressing
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by A are added to or subtracted from the contents of the location specified by B and placed in the accumulator. [1]
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by A are added to or subtracted from the contents of the location specified by B and delivered to the location specified by C.

[1] Hunting is inhibited.

Table 5-5. Masked Inactive Addressing for Add and Subtract Instructions

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	No operation takes place. [1]
INACTIVE	INACTIVE	ACTIVE	The masked contents of the accumulator are delivered to the location specified by C, which is protected.
INACTIVE	ACTIVE	INACTIVE	The contents of the accumulator are added to or subtracted from the masked contents of the location specified by B; the result is placed in the accumulator. The contents of the mask are delivered to the mask register. [1]
INACTIVE	ACTIVE	ACTIVE	The contents of the accumulator are added to or subtracted from the masked contents of the location specified by B. The masked result is delivered to the location specified by C, which is protected.
ACTIVE	INACTIVE	INACTIVE	The masked contents of the location specified by A are placed in the accumulator. The contents of the mask are delivered to the mask register. [1]
ACTIVE	INACTIVE	ACTIVE	The masked contents of the location specified by A are delivered to the location specified by C, which is protected.
ACTIVE	ACTIVE	INACTIVE	The masked contents of the location specified by A are added to or subtracted from the masked contents of the location specified by B and the result is placed in the accumulator. The contents of the mask are delivered to the mask register. [1]
ACTIVE	ACTIVE	ACTIVE	The masked contents of the location specified by A are added to or subtracted from the masked contents of the location specified by B. The masked result is delivered to the location specified by C, which is protected.

[1] Hunting is inhibited.

Table 5-6. Unmasked Inactive Addressing for Accumulate Instructions

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	No operation takes place. [1]
INACTIVE	INACTIVE	ACTIVE	When B is void, the behavior of the system is unspecified.
INACTIVE	ACTIVE	INACTIVE	The contents of the accumulator are totaled the number of times specified by the high-order six bits of the B address. The low-order results are stored with the proper sign in the accumulator. [1]
INACTIVE	ACTIVE	ACTIVE	The contents of the accumulator are totaled the number of times specified by the high-order six bits of the B address. The low-order results are stored with the proper sign in C.
ACTIVE	INACTIVE	INACTIVE	When B is void, the behavior of the system is unspecified. [1]
ACTIVE	INACTIVE	ACTIVE	When B is void, the behavior of the system is unspecified.
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by A are added the number of times specified by the high-order six bits of the B address. The low-order results are stored with the proper sign in the accumulator. [1]
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by A are added the number of times specified by the high-order six bits of the B address. The low-order results are stored with the proper sign in C.

[1] Hunting is inhibited.

Table 5-7. Unmasked Inactive Addressing for Multiply Instructions

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	No operation takes place. [1]
INACTIVE	INACTIVE	ACTIVE	The contents of the accumulator are multiplied by themselves and the high-order product is stored with the proper sign in C; the low-order product is stored with the proper sign in the low-order product register.
INACTIVE	ACTIVE	INACTIVE	The contents of the accumulator are multiplied by the absolute value of the contents of B. The high-order product is stored with the proper sign in the accumulator; the low-order product is stored with the proper sign in the low-order product register. The submultiple 1 is delivered to location 1 (program address) in main memory with bad parity (for Binary Multiply only). [1]

Table 5-7 (cont). Unmasked Inactive Addressing for Multiply Instructions

A	B	C	Action for void addressing
INACTIVE	ACTIVE	ACTIVE	The contents of the accumulator are multiplied by the absolute value of the contents of B. The high-order product is stored with the proper sign in C; the low-order product is stored with the proper sign in the low-order product register.
ACTIVE	INACTIVE	INACTIVE	The contents of the location specified by A are multiplied by the contents of the accumulator. The high-order product is stored with the proper sign in the accumulator; the low-order product is stored with the proper sign in the low-order product register. The submultiple 1 is delivered to location 1 (program address) in main memory with bad parity (for Binary Multiply only). [1]
ACTIVE	INACTIVE	ACTIVE	The contents of the location specified by A are multiplied by the contents of the accumulator. The high-order product C; the low-order product is stored with the proper sign in the low-order product register.
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by A are multiplied by the contents of the location specified by B. The high-order product is stored with the proper sign in the accumulator; the low-order product is stored with the proper sign in the low-order product register. The submultiple 1 is delivered to location 1 (program address) in main memory with bad parity (for Binary Multiply only). [1]
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by A are multiplied by the contents of the location specified by B. The high-order product is stored with the proper sign in both the accumulator and C; the low-order product is stored with proper sign in the low-order product register.

[1] Hunting is inhibited.

Table 5-8. Unmasked Inactive Addressing for Divide Instructions

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	A division overcapacity unprogrammed transfer to U + 10 or U + 11 occurs. [1]
INACTIVE	INACTIVE	ACTIVE	A division overcapacity unprogrammed transfer to U + 10 or U + 11 occurs.
INACTIVE	ACTIVE	INACTIVE	The contents of the location specified by B are divided by the contents of the accumulator; the quotient is retained in the accumulator and the remainder in the low-product register. [1]

Table 5-8 (cont). Unmasked Inactive Addressing for Divide Instructions

A	B	C	Action for void addressing
INACTIVE	ACTIVE	ACTIVE	The contents of the location specified by B are divided by the contents of the accumulator; the quotient is retained in the location specified by C and the remainder in the low-order product register.
ACTIVE	INACTIVE	INACTIVE	The contents of the accumulator are divided by the contents of the location specified by A; the quotient is retained in the accumulator and the remainder in the low-order product register.[1]
ACTIVE	INACTIVE	ACTIVE	The contents of the accumulator are divided by the contents of the location specified by A; the quotient is retained in the location specified by C and the remainder in the low-order product register.
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by B are divided by the contents of the location specified by A; the quotient is retained in the accumulator and the remainder in the low-order product register.[1]
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by B are divided by the contents of the location specified by A; the quotient is retained in the location specified by C and the remainder in the low-order product register.

[1] Hunting is inhibited.

TRANSFER INSTRUCTIONS

The logic of the word processor includes seven instructions designed to transfer data within the main memory, within the control memory, or from one memory to another. Using these instructions, called transfer instructions, the programmer may move any desired quantity of information, from a single bit to an entire record. Two of the instructions move single words only, or when masked, fields within single words. The other five instructions move groups of words and cannot be masked.

The command codes for all seven instructions provide the ability to address control registers and to designate either sequencing counter as the source of the next instruction. In instructions which transfer groups of words, the A-address field indicates the location of the first word to be transferred, while the C-address field designates the location to which this word is to be delivered. Except for the Multiple Transfer (MT) instructions, transfers involving groups of words occur under control of control registers AU1 and AU2 which initially contain one of the following bit configurations, depending upon the type of addressing used in the A- and C-address fields, respectively.

Normal Mode

1. Direct Memory Location Address: A positive sign bit, the array and bank indicator bits taken from the sequencing counter which selected the instruction, and the low-order 11 bits of the address field.
2. Direct Control Register Address: A positive sign bit, the group indicator, and the low-order 11 bits of the address field (bits 2 through 9 are automatically changed to zeros).
3. Indexed Memory Location Address: A positive sign bit and the augmented low-order 23 bits of the index register specified in the address field.
4. Index Control Register Address: A positive sign bit and the augmented low-order 23 bits of the index register specified in the address field.
5. Indirect Memory Location Address: The sign bit (positive or negative) and 23-bit main memory address stored in the control register which was designated in the address field.
6. Indexed Indirect Memory Location Address: The sign bit (positive or negative) and 23-bit main memory address stored in the control register whose address is obtained by augmenting the contents of the index register specified in the address field.

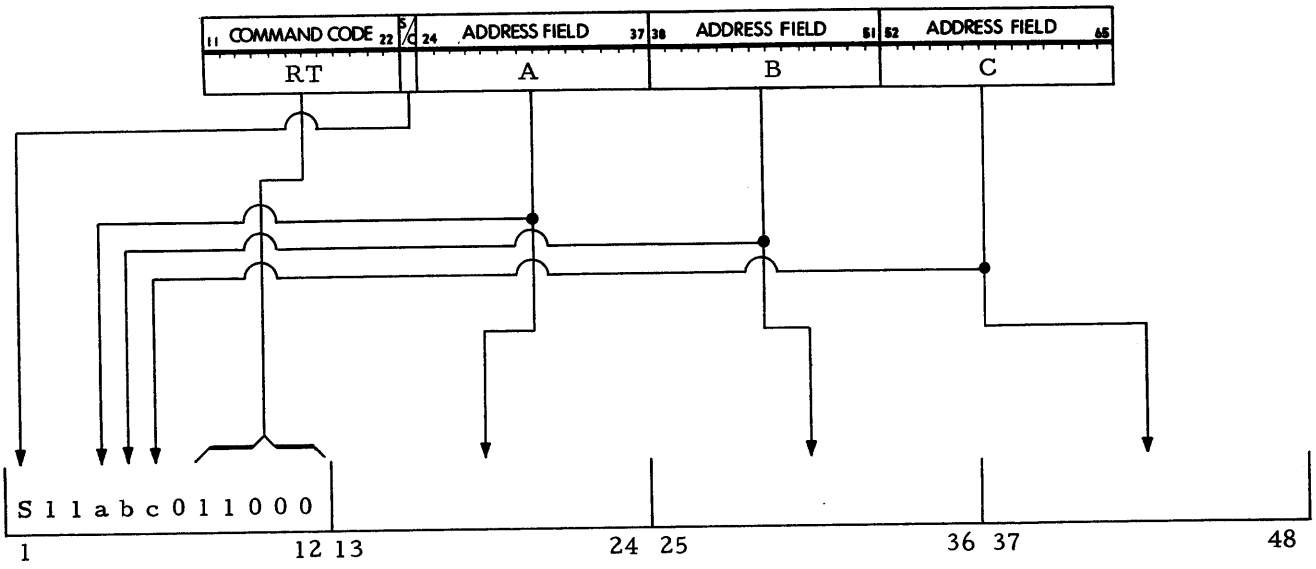
Extended Mode

1. Direct Memory Location Address: A positive sign bit and the low-order 23 bits of the address field.
2. Direct Control Register Address: A positive sign bit, group indicator, and the low-order 11 bits of the address field. Bits 2 through 9 are automatically set to zeros. (Bits 2 through 18 of the address field must be all zeros.)
3. Indexed Memory Location Address: A positive sign bit and the augmented low-order 23 bits of the index register specified in the address field.
4. Indexed Control Register Address: A positive sign bit and the augmented low-order 23 bits of the index register specified in the address field.
5. Indirect Memory Location Address: The sign bit (positive or negative) and 23-bit main memory address stored in the control register designated in the address field.
6. Indexed Indirect Memory Location Address: The sign bit (positive or negative) and the 23-bit main memory address stored in the control register whose address is obtained by augmenting the contents of the index register specified in the address field.

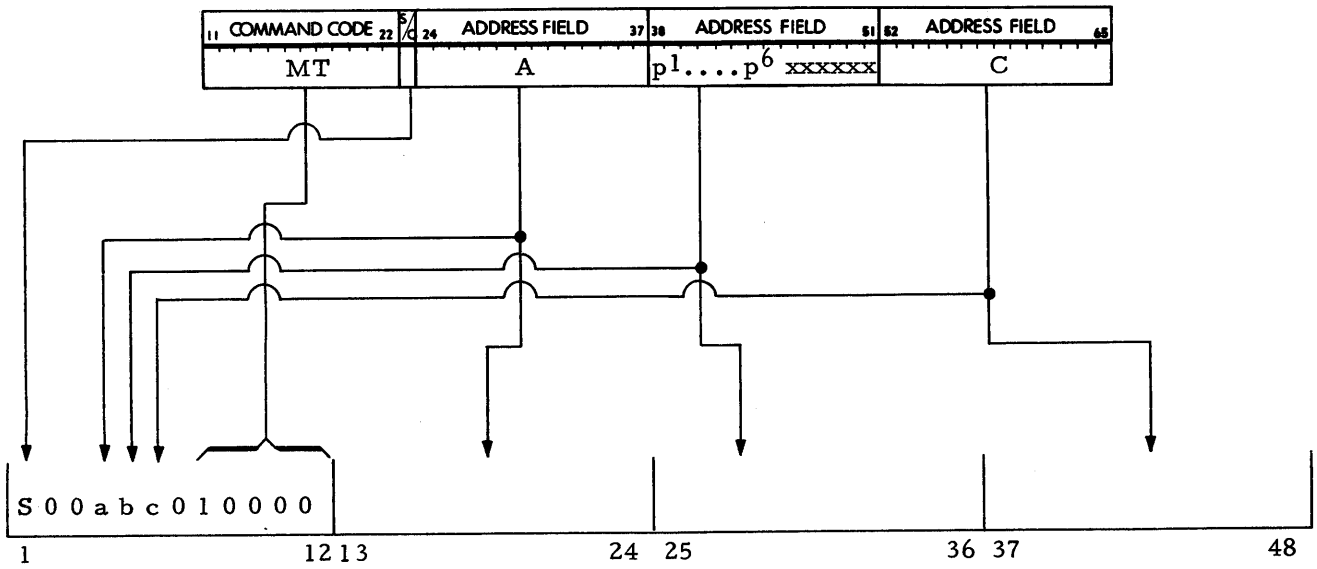
As successive words are transferred, the arithmetic control counters are automatically incremented (or decremented if the sign bit is negative) by one to specify a source and a result address for each word transferred. At the completion of the instruction, the counters contain addresses equal to their initial settings plus (or minus) the number of words transferred, i.e., the address of the last word transferred plus (or minus) one.

Assembly

The Transfer and Sequence Change (TS), Record Transfer (RT), and Item Transfer (IT) instructions in the transfer grouping are assembled as indicated in the following diagram (using the Record Transfer instruction).



The following diagram (using the Multiple Transfer instruction) displays the instruction assembly for the Multiple Transfer (MT), N-Word Transfer (TN), and Extended N-Word Transfer (ETN) instructions. Variation from the above example occurs in the B-address field which, in these instances, contains parameters



The B-address field in the Transfer A to C instruction (TX) is not used in the instruction assembly.

TX | TRANSFER A TO C

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B10 abc 010 000 $(2020)_8$	A	(B) not used	C

FUNCTION

The Transfer A to C instruction transfers one word from the location specified by the A-address field to the location specified by the C-address field. The B-address field is ignored. When the instruction is used with a mask, partial words (or fields) may be transferred from one memory location to another, protecting the unmasked portion of the result location.

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 110 000 $(0060)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D, I	not used	D, I
Unmasked	All	not used	All

Inactive:

Both masked and unmasked versions of the Transfer A to C instruction may use inactive addressing in the manner described in Tables 5-9 and 5-10, on page 5-65.

TS | TRANSFER AND SEQUENCE CHANGE

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B10 abc 000 100 $(2004)_8$	A	B	C

FUNCTION

The Transferred Sequence Change instruction transfers one word from the location specified by the A-address field to the location designated by the B-address field

and changes the setting of the specified sequencing counter so that the next instruction is selected from the main memory location designated by the C-address field. The appropriate history register is changed after transfer of information from A to B but before the setting of the sequencing counter is changed. The instruction is used, with inactive addressing, to provide access to the low-order product register.

The A and B addresses may use any of the six types of addressing previously discussed for both normal and extended modes. If the C address is active, it may specify a direct memory location address, an indirect memory location address, an indexed memory location address, or an indexed indirect memory location address in either the normal or the extended mode. The behavior of the system for each of these cases is described below.

Normal Mode

1. Direct Memory Location Address. The low-order 11 bits of the C-address field are placed in the specified sequencing counter, together with the bank indicator and array bits from the counter which selected the instruction. The sign bit is set to a plus value.

2. Indirect Memory Location Address. The complete contents of the addressed control register, including the sign, bank indicator, and array bits are transferred to the specified sequencing counter. The contents of the addressed control register are incremented in the usual fashion after use, unless the control register addressed is the counter to be changed (either sequence or cosequence counter), in which case incrementing does not take place. The tabular bit in the address field is ignored.

3. Indexed Memory Location Address. The contents of the referenced index register are augmented, and the complete 23-bit address thus formed is inserted, with a positive sign, into the specified sequencing counter.

4. Indexed Indirect Memory Location Address. A complete control register address is generated by augmenting the contents of the referenced index register in the usual fashion. The generated address specifies a control register whose entire 24-bit contents are transferred to the specified sequencing counter. The contents of the addressed control register are incremented in the usual fashion after use, unless this control register is the counter to be changed (either sequence or cosequence counter). The tabular bit in the address field is ignored.

Extended Mode

1. Direct Memory Location Address. The low-order 23-bits of the C-address field, together with a positive sign bit, are loaded directly into the specified sequencing counter.

2. Indirect Memory Location Address. The unincremented 23-bit contents, plus sign, of the specified control register are loaded into the specified sequencing counter. Incrementing occurs as usual, unless the specified control register is the counter to be changed. The tabular bit in the address field is ignored.

3. Indexed Memory Location Address. The augmented 23-bit contents, plus sign, of the specified control register are loaded into the specified sequencing counter.

4. Indexed Indirect Memory Location Address. The augmented 23-bit contents, plus sign, of the specified control register are loaded into the specified sequencing counter. Incrementing occurs as usual, unless the specified control register is the counter to be changed. The tabular bit in the address field is ignored.

If a parity error is detected in a word selected from main memory during execution of this instruction, an unprogrammed transfer occurs and the contents of the specified sequencing counter are not changed. The contents of the appropriate history register will be changed, however, and the bisequence bit in the program control register will be set in response to the bisequence bit in the command code of the instruction.

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 100 100 $(0044)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D, I	D, I	D, I
Unmasked	All	All	D, I, H, IH

Inactive:

Both masked and unmasked versions of the Transfer and Sequence Change instruction may use inactive addressing in the manner described in Tables 5-11 and 5-12, on pages 5-65 and 5-66.

TN | N-WORD TRANSFER

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B01 abc 010 000 $(1020)_8$	A	$p^1 \ldots p^6$ xxxxxx	C

FUNCTION

The N-Word Transfer instruction transfers the number of words specified by the high-order six bits of the B-address field from consecutive locations starting at A to consecutive locations starting at C. The number of words to be transferred can range from 0 to 63. If the B-address field is zero, no information is transferred. The low-order six bits of B are ignored. The transfer of information occurs under control of control registers AU1 and AU2, according to the conventions outlined at the beginning of this grouping of instructions.

It should be noted that if a control register is directly addressed in the A address of an N-Word Transfer instruction and an increment other than zero appears in the address field, then this increment is applied to the register after each transfer.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	p	All

Inactive:

The N-Word Transfer instruction may use inactive addressing in the manner described in Table 5-13, on page 5-67.

ETN | EXTENDED N-WORD TRANSFER

FORMAT

1 12	13 24	25 36	37 48
Command Code	Address Field	Address Field	Address Field
B00 abc 010 100 $(0024)_8$	A	$p^1 \ldots . p^6$ xxxxxx	C

FUNCTION

The Extended N-Word Transfer instruction performs the same function as the N-Word Transfer (TN) instruction, with the following exceptions:

1. When transferring words from one set of main memory locations to another set of main memory locations, 16 punctuation bits of each word are moved in addition to the 48 data bits.
2. When transferring the contents from control memory registers to locations defined by the C-address field (in main memory), all 24 data bits of the register are moved. However, in contrast to the TN instruction, the high-order 24 data bits and the backround punctuation bits of each C address are preserved. If the B memory-designator bit is a zero, it indicates that the sign (one bit) will be transferred with the 23-bit address/data; if the B memory-designator bit is a one, the address/data is transferred as usual, but the sign-bit portion in the location specified by C is cleared to zero.
3. When transferring the contents from control memory locations to control memory, all 24 bits of the register are moved.
4. When transferring the contents from main memory to control memory, 24 bits (25 through 48) of main memory locations are placed in control memory.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	p	All

Inactive:

The Extended N-Word Transfer instruction may use inactive addressing in the manner described in Table 5-13, on page 5-67.

MT | MULTIPLE TRANSFER

FORMAT

1 12 13 24 25 36 37 48

Command Code	Address Field	Address Field	Address Field
B00 abc 010 000 $(0020)_8$	A	$p^1 \ldots . p^6$ xxxxxx	C

FUNCTION

The Multiple Transfer instruction transfers the contents of the location specified by the A-address field to the location specified by the C-address field, repeating the transfer the number of times specified by the high-order six bits of the B-address field. This number may range from 0 to 63. If B is zero, no transfer of information takes place. The low-order six bits of B are ignored.

Although all types of addressing are permitted with this instruction, the instruction is most meaningful when used with indirect addressing with increment.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	p	All

Inactive:

The Multiple Transfer instruction may use inactive addressing in the manner described in Table 5-13, on page 5-67.

RT | RECORD TRANSFER

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B11 abc 011 000 $(3030)_8$	A	B	C

FUNCTION

The Record Transfer instruction is used to move a group of related words from one location to another. Such a group does not necessarily constitute a record. Although the Record Transfer instruction is most frequently used to manipulate "records," it may also be used to advantage whenever the number of words to be transferred is greater than 63, the maximum number which can be moved with the N-Word Transfer. In fact, the only restriction on the number of words which can be transferred is the practical limitation of available storage space in the memory.

When a Record Transfer instruction is executed, an end-of-record word[1] is stored in the location specified by the B-address field. Consecutive words are then transferred from the location starting with A to consecutive locations starting with C, until an end-of-record word is transferred.

Note that the transfer of information is stopped whenever an end-of-record is transferred, regardless of where this word was stored. In other words, the word stopping the transfer is not necessarily the same word which the instruction has stored in the location specified by the B address. The transfer also stops if a parity failure occurs. At the completion of the instruction, AU1 will contain the complete address generated from the A-address field plus the number of words actually transferred; AU2 will contain the complete address generated from the C-address field plus the number of words transferred.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	D, I, H, IH	All	D, I, H, IH

Inactive:

The Record Transfer instruction may use inactive addressing in the manner described in Table 5-14, on page 5-68.

[1] An end-of-record word is a word whose 48 information bits are:
1010 1010 0000 0000 1110 1110 1110 1101 1101 1101 1101 (hexadecimally, BBOOFFFFEEEE).

IT | ITEM TRANSFER

FORMAT

1 – 12	13 – 24	25 – 36	37 – 48
Command Code	Address Field	Address Field	Address Field
B01 abc 011 000 $(1030)_8$	A	B	C

FUNCTION

The Item Transfer instruction is used to move a group of related words from one location to another. When an Item Transfer instruction is executed, an end-of-item symbol[1] is substituted for the high-order 32 bits of the contents of B, clearing the low-order 16 bits of B to all zeros. Consecutive words are transferred from the location starting with A to consecutive locations starting with C, until an end-of-item symbol is transferred.

Note that the transfer of information stops with the transfer of an end-of-item or end-of-record word, or with a parity failure. At the completion of the instruction, AU1 will contain the complete address generated from the A-address field, plus the number of words actually transferred; AU2 will contain the complete address generated from the C-address field plus the number of words transferred.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	D, I, H, IH	All	D, I, H, IH

Inactive:

The Item Transfer instruction may use inactive addressing in the manner described in Table 5-14, on page 5-68.

[1] An end-of-item symbol is a word whos high-order 32 bits are:
1010 1010 0000 0000 1110 1110 1110 1110 (hexadecimally, BBOOFFFF).

Table 5-9. Unmasked Inactive Addressing for the Transfer A to C (TX) Instruction

A	B	C	Action for void addressing
INACTIVE	X	INACTIVE	No operation takes place. [1]
INACTIVE	X	ACTIVE	The contents of the accumulator are delivered to the location specified by C.
ACTIVE	X	INACTIVE	The contents of the location specified by A are placed in the accumulator. [1]
ACTIVE	X	ACTIVE	The contents of the location specified by A are placed in the accumulator and delivered to the location specified by C.

X = irrelevant

[1] Hunting is inhibited.

Table 5-10. Masked Inactive Addressing for the Transfer A to C (TX) Instruction

A	B	C	Action for void addressing
INACTIVE	X	INACTIVE	No operation takes place. [1]
INACTIVE	X	ACTIVE	The masked contents of the accumulator are delivered to the location specified by C protecting the unmasked portion of the result location.
ACTIVE	X	INACTIVE	The masked contents of the location specified by A are placed in the accumulator. [1]
ACTIVE	X	ACTIVE	The masked contents of the location specified by A are placed in the accumulator and are delivered to the location specified by C, protecting the unmasked portion of the result location.

X = irrelevant

[1] Hunting is inhibited.

Table 5-11. Unmasked Inactive Addressing for the Transfer and Sequence Change (TS) Instruction

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	No operation takes place. [1]
INACTIVE	INACTIVE	ACTIVE	The setting of the specified sequencing counter is changed so that the next instruction is selected from the main memory location designated by the C-address field.
INACTIVE	ACTIVE	INACTIVE	The contents of the accumulator are delivered to the location specified by the B-address field. [1]
INACTIVE	ACTIVE	ACTIVE	The contents of the accumulator are delivered to the location specified by the B-address field, and the setting of the specified sequencing counter is

Table 5-11 (cont). Unmasked Inactive Addressing for the Transfer and Sequence Change (TS) Instruction

A	B	C	Action for void addressing
INACTIVE	ACTIVE	ACTIVE	changed so that the next instruction is selected from the main memory location designated by the C-address field.
ACTIVE	INACTIVE	INACTIVE	The contents of the location specified by the A-address field are placed in the accumulator. [1]
ACTIVE	INACTIVE	ACTIVE	The contents of the location specified by the A-address field are placed in the accumulator. The setting of the specified sequencing counter is changed so that the next instruction is selected from the main memory location designated by the C-address field.
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by the A-address field are placed in the accumulator and delivered to the location specified by the B-address field. [1]
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by the A-address field are placed in the accumulator and delivered to the location specified by the B-address field. The setting of the specified sequencing counter is changed so that the next instruction is selected from the main memory location designated by the C-address field.

[1] Hunting is inhibited.

Table 5-12. Masked Inactive Addressing for the Transfer and Sequence Change (TS) Instruction

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	No operation takes place. [1]
INACTIVE	INACTIVE	ACTIVE	The setting of the specified sequencing counter is changed so that the next instruction is selected from the main memory location designated by the C-address field.
INACTIVE	ACTIVE	INACTIVE	The masked contents of the accumulator are delivered to the location specified by B, protecting the unmasked portion of the result location. [1]
INACTIVE	ACTIVE	ACTIVE	The masked contents of the accumulator are delivered to the location specified by B, protecting the unmasked portion of the result location. The setting of the specified sequencing counter is changed so that the next instruction is selected from the main memory location designated by the C-address field.

Table 5-12 (cont). Masked Inactive Addressing for the Transfer and Sequence Change (TS) Instruction

A	B	C	Action for void addressing
ACTIVE	INACTIVE	INACTIVE	The masked contents of the location specified by the A-address field are placed in the accumulator.[1]
ACTIVE	INACTIVE	ACTIVE	The masked contents of the location specified by the A-address field are placed in the accumulator. The setting of the specified sequencing counter is changed so that the next instruction is selected from the main memory location designated by the C-address field.
ACTIVE	ACTIVE	INACTIVE	The masked contents of the location specified by the A-address field are placed in the accumulator and are delivered to the location specified by B, protecting the unmasked portion of the result location. The setting of the specified sequencing counter changes so that the next instruction is selected from the main memory location designated by the C-address field. [1]

[1] Hunting is inhibited.

Table 5-13. Unmasked Inactive Addressing for the Multiple Transfer (MT), N-Word Transfer (TN), and Extended N-Word Transfer (ETN) Instructions

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	No operation takes place. [1]
INACTIVE	INACTIVE	ACTIVE	When B is void, the behavior of the system is unspecified.
INACTIVE	ACTIVE	INACTIVE	The contents of the accumulator are transferred to the locations specified by C the number of times specified by the high-order six bits of the B-address. The low-order results are stored in the accumulator. [1]
INACTIVE	ACTIVE	ACTIVE	The contents of the accumulator are transferred to the locations specified by C the number of times specified by the high-order six bits of the B-address. The low-order results are stored in the location specified by C.
ACTIVE	INACTIVE	INACTIVE	When B is void, the behavior of the system is unspecified. [1]
ACTIVE	INACTIVE	ACTIVE	When B is void, the behavior of the system is unspecified.
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by A are transferred to the locations specified by C the number of times specified by the high-order six bits of the B-address. The low-order results are stored in the accumulator. [1]

Table 5-13 (cont). Unmasked Inactive Addressing for the Multiple Transfer (MT), N-Word Transfer (TN), and Extended N-Word Transfer (ETN) Instructions

A	B	C	Action for void addressing
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by A are transferred to the locations specified by C the number of times specified by the high-order six bits of the B-address. The low-order results are stored in the location specified by C.

[1]Hunting is inhibited.

Table 5-14. Unmasked Inactive Addressing for the Record Transfer (RT) and Item Transfer (IT) Instructions

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	The instruction is trapped as having an illegal command code; this results in a call to master control.
INACTIVE	INACTIVE	ACTIVE	The instruction is trapped as having an illegal command code; this results in a call to master control.
INACTIVE	ACTIVE	INACTIVE	The instruction is trapped as having an illegal command code; this results in a call to master control.
INACTIVE	ACTIVE	ACTIVE	The instruction is trapped as having an illegal command code; this results in a call to master control.
ACTIVE	INACTIVE	INACTIVE	The action of the system is unspecified. [1]
ACTIVE	INACTIVE	ACTIVE	The action of the system is unspecified.
ACTIVE	ACTIVE	INACTIVE	The action of the system is unspecified. [1]
ACTIVE	ACTIVE	ACTIVE	(Item Transfer) Substitute end-of-item symbol for the high-order 32 bits of B, clearing the low-order 16 bits of B to all zeros. Transfer words from consecutive locations starting with C until an end-of-item symbol is transferred. Upon completion, AU1 contains A + n, AU2 contains C + n.
			(Record Transfer) Stores end-of-record word in B. Transfers words from consecutive locations starting with A to consecutive locations starting with C until an end-of-record word is transferred. Otherwise, same as Item Transfer.

[1]Hunting is inhibited.

DECISION INSTRUCTIONS

The word processor logic includes four decision instructions which are used to branch to

an alternate path of control in response to a precisely defined condition. Each of these instructions looks for a special condition based upon the relationship between the contents of the location specified by the A-address field and the contents of the location specified by the B-address field. If the condition is met, the sequencing counter designated as the source of the next instruction is changed so that the next instruction is selected from the main memory location specified by the C-address field. If the condition fails, the current setting of the designated sequencing counter selects the next instruction.

Any type of addressing may be used in the A- and B-address fields of these instructions unless they are masked. The C-address field may contain a direct memory location address, an indirect memory location address, an indexed memory location address, or an indexed indirect memory location address. If the condition is met and the sequencing counter is changed, the behavior of the system for each of the four possible types of C address is the same as that described for the Transfer and Sequence Change instruction (page 5-58). If the C address is inactive, no change is made in either sequencing counter and hunting for the next program in demand is inhibited. If the C address is active and the condition is met, hunting is also inhibited. If the condition is not met, a normal hunt is made for the next program in demand.

If a parity error is detected in a word selected from main memory during execution of a decision instruction, an unprogrammed transfer is executed to U + 2 or U + 3, and the contents of the sequencing counter are not altered. However, the contents of the appropriate history register may be changed: if the condition is met, the history register will be changed; if the condition fails, the history register will not be changed. In either case, the bisequence bit in the program control register is set in response to the bisequence bit in the command code of the instruction.

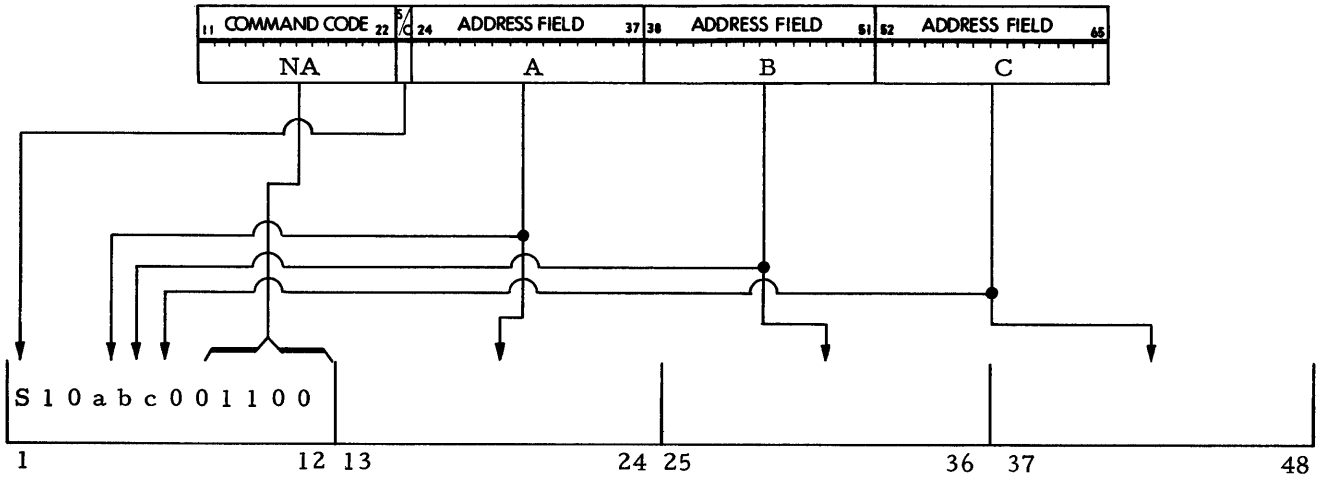
Two of the four decision instructions are inequality comparisons (one alphabetic, one numeric) in which the two operands are examined for inequality ($\neq$). If they are not equal, the next instruction is selected from the location specified by the C-address field. The other two instructions (one alphabetic, one numeric) are called "less than" comparisons, since the two operands are inspected to determine whether the contents of A are less than or equal to the contents of B ($\leq$). If so, the next instruction is selected from the location specified by C. All four instructions may compare either the entire contents of both words (unmasked version) or respective fields of the words, as defined by a mask. The location of the mask, if any, is partially specified by bits of the command code.

The alphabetic comparisons make a bit-by-bit comparison of all 48 information bits of the words at locations A and B. If the word at A has a zero bit in the first position where they differ, reading from left to right, it is considered "alphabetically less" than the word at B. When the instructions are masked, the decision is based upon a bit-by-bit comparison of the selected fields of the operands.

The numeric comparisons treat both operands as signed 11-digit numbers and make a bit-by-bit comparison of information bits 5 through 48 of the contents of A and B. Thus, the 11 digits of the signed numeric words are compared as in the alphabetic comparisons. However, the sign bits (bits 1 through 4) of both words are checked before a final decision is made. If any of the four bits is a one, the sign is considered to be positive; if all four bits are zeros, the sign is negative. Any positive number is larger than any negative number, with the exception that two words with 11 low-order zero digits are considered equal regardless of sign. When two negative numbers are compared, the one with absolute value closer to zero is considered greater. Thus, the numeric comparisons are true algebraic comparisons of signed 11-digit numbers.

Assembly

All four instructions in the decision grouping are assembled as indicated in the following diagram (using the Inequality Comparison, Alphabetic instruction):



NA | INEQUALITY COMPARISON, ALPHABETIC

FORMAT

1 – 12	13 – 24	25 – 36	37 – 48
Command Code	Address Field	Address Field	Address Field
B10 abc 001 100 $(2014)_8$	A	B	C

FUNCTION

The Inequality Comparison, Alphabetic instruction compares for inequality the 48-bit contents of the word at A and the 48-bit contents of the word at B. If the two words are not identical, the sequencing counter specified as the source of the next instruction is changed so that the next instruction is selected from the location designated by the C-address field. If they are equal, the next instruction is selected from the location specified by the current contents of the designated sequencing counter. Plus zero is not equal to minus zero.

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 101 100 $(0054)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D, I	D, I	D, I
Unmasked	All	All	D, I, IH, H

Inactive:

Both masked and unmasked versions of the Inequality Comparison, Alphabetic instruction may use inactive addressing in the manner described in Tables 5-15 and 5-17 on pages 5-74 and 5-75.

LA LESS THAN OR EQUAL COMPARISON, ALPHABETIC

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B10 abc 011 100 $(2034)_8$	A	B	C

FUNCTION

The Less Than or Equal Comparison, Alphabetic instruction makes a bit-by-bit comparison of the 48-bit word at A and the 48-bit word at B. If the contents of the location specified by the A address are less than or equal to the contents of the location specified by B, the designated sequencing counter is changed so that the next instruction is selected from the location specified by the C-address field. If the word at address A is greater than the word at address B, the next instruction is taken from the location specified by the current contents of the designated sequencing counter. Plus zero is greater than minus zero.

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 111 100 $(0074)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D, I	D, I	D, I
Unmasked	All	All	D, I, IH, H

Inactive:

Both masked and unmasked versions of the Less Than or Equal Comparison, Alphabetic instruction may use inactive addressing in the manner described in Tables 5-16 and 5-18, on pages 5-74 and 5-75.

NN | INEQUALITY COMPARISON, NUMERIC

FORMAT

1 12	13 24	25 36	37 48
Command Code	Address Field	Address Field	Address Field
B10 abc 001 000 $(2010)_8$	A	B	C

FUNCTION

The Inequality Comparison, Numeric instruction examines the contents of the locations specified by A and B, regarded as signed words, to determine whether they are algebraically unequal. If they are unequal, the designated sequencing counter is changed to select the next instruction from the location specified by the C-address field. If the words are equal, the address of the next instruction is selected from the location specified by the current contents of the designated sequencing counter. In a numeric inequality comparison, plus zero equals minus zero.

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 101 000 $(0050)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D, I	D, I	D, I
Unmasked	All	All	D, I, IH, H

Inactive:

Both masked and unmasked versions of the Inequality Comparison, Numeric instruction may use inactive addressing in the manner described in Tables 5-15 and 5-17, on pages 5-74 and 5-75.

LN | LESS THAN OR EQUAL COMPARISON, NUMERIC

FORMAT

1 12 13 24 25 36 37 48

Command Code	Address Field	Address Field	Address Field
B10 abc 011 000 $(2030)_8$	A	B	C

FUNCTION

When the Less Than or Equal Comparison, Numeric instruction is executed, the contents of the locations specified by the A- and B-address fields are regarded as signed words and compared to determine whether the word at A is algebraically less than or equal to the word at B. If the A operand is less than or equal to the B operand, the designated sequencing counter is changed to select the next instruction from the location specified by the C-address field. If the contents of A are algebraically greater than the contents of B, the next instruction is selected from the location specified by the current contents of the designated sequencing counter. Plus zero equals minus zero in a numeric less than or equal comparison.

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 111 000 $(0070)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D, I	D, I	D, I
Unmasked	All	All	D, I, IH, H

Inactive:

Both masked and unmasked versions of the Less Than or Equal Comparison, Numeric instruction may use inactive addressing in the manner described in Tables 5-16 and 5-18, on pages 5-74 and 5-75.

Table 5-15. Unmasked Inactive Addressing for the Inequality Comparison, Alphabetic (NA), Inequality Comparison, Numeric (NN), and Normalized Inequality Comparison (FNN) Instructions

A	B	C	Action for void addressing
X	X	INACTIVE	No operation takes place.
INACTIVE	INACTIVE	ACTIVE	Compares the contents of the accumulator for inequality to zero. [1]
INACTIVE	ACTIVE	ACTIVE	Compares the contents of the accumulator for inequality to the contents of the location specified by B. [1]
ACTIVE	INACTIVE	ACTIVE	Compares the contents of the location specified by A for inequality to zero. [1]
ACTIVE	ACTIVE	ACTIVE	Compares the contents of the location specified by A for inequality to the contents of the location specified by B. [1]

X = irrelevant

[1] If unequal, then the sequencing counter specified as the source of the next instruction is changed so that the next instruction is selected from the location designated by the C-address field.
If equal, the next instruction is selected from the location specified by the current contents of the designated sequencing counter.

Table 5-16. Unmasked Inactive Address for the Less Than or Equal Comparison, Alphabetic (LA), Less Than or Equal Comparison, Numeric (LN), and the Normalized Less Than Comparison (FLN) Instructions

A	B	C	Action for void addressing
X	X	INACTIVE	No operation takes place.
INACTIVE	INACTIVE	ACTIVE	Compares the contents of the accumulator for less than or equality to zero. [1]
INACTIVE	ACTIVE	ACTIVE	Compares the contents of the accumulator for less than or equality to the contents of the location specified by B. [1]
ACTIVE	INACTIVE	ACTIVE	Compares the contents of the location specified by A for less than or equality to zero. [1]
ACTIVE	ACTIVE	ACTIVE	Compares the contents of the location specified by A for less than or equality to the contents of the location specified by B. [1]

X = irrelevant

[1] If less than or equal to zero, the designated sequencing counter is changed so that the next instruction is selected from the location specified by the C-address field.
If greater than zero, the next instruction is taken from the location specified by the current contents of the designated sequencing counter.

Table 5-17. Masked Inactive Addressing for the Inequality Comparison, Alphabetic (NA) and Inequality Comparison, Numeric (NN) Instructions

A	B	C	Action for void addressing
X	X	INACTIVE	The contents of the mask are delivered to the mask register.
INACTIVE	INACTIVE	ACTIVE	The contents of the mask are delivered to the mask register. Compares the contents of the accumulator for inequality to zero.[1]
INACTIVE	ACTIVE	ACTIVE	The contents of the mask are delivered to the mask register. Compares the contents of the accumulator for inequality to the masked contents of the location specified by B.[1]
ACTIVE	INACTIVE	ACTIVE	The contents of the mask are delivered to the mask register. Compares the masked contents of the location specified by A for inequality to zero.[1]
ACTIVE	ACTIVE	ACTIVE	The contents of the mask are delivered to the mask register. Compares the masked contents of the location specified by A for inequality to the masked contents of the location specified by B.[1]

X = irrelevant

[1] If unequal, then the sequencing counter specified as the source of the next instruction is changed so that the next instruction is selected from the location designated by the C-address field.
If equal, the next instruction is selected from the location specified by the current contents of the designated sequencing counter.

Table 5-18. Masked Inactive Addressing for the Less Than or Equal Comparison, Alphabetic (LA) and Less Than or Equal Comparison, Numeric (LN) Instructions

A	B	C	Action for void addressing
X	X	INACTIVE	The contents of the mask are delivered to the mask register.
INACTIVE	INACTIVE	ACTIVE	The contents of the mask are delivered to the mask register. Compares the contents of the accumulator for less than or equality to zero.[1]
INACTIVE	ACTIVE	ACTIVE	The contents of the mask are delivered to the mask register. Compares the contents of the accumulator for less than or equality to the masked contents of the location specified by B.[1]
ACTIVE	INACTIVE	ACTIVE	The contents of the mask are delivered to the mask register. Compares the masked contents of the location specified by A for less than or equality to zero.[1]
ACTIVE	ACTIVE	ACTIVE	The contents of the mask are delivered to the mask register. Compares the masked contents

Table 5-18 (cont). Masked Inactive Addressing for the Less Than or Equal Comparison, Alphabetic (LA) and Less Than or Equal Comparison, Numeric (LN) Instructions

A	B	C	Action for void addressing
ACTIVE	ACTIVE	ACTIVE	of the location specified by A for less than or equality to the masked contents of the location specified by B.[1]

X = irrelevant

[1] If less than or equal to zero, the designated sequencing counter is changed so that the next instruction is selected from the location specified by the C-address field. If greater than zero, the next instruction is taken from the location specified by the current contents of the designated sequencing counter.

SHIFT INSTRUCTIONS

The ability to pack as many as four separate pieces of signed information (or a greater number of pieces of unsigned information) into a single Honeywell 8200 word would have little practical value without the facility for manipulating each piece of information (or field) independently. In part, this facility is provided by the masking feature of the system, which permits certain instructions to operate on fields in preassigned positions within words. In order to achieve maximum flexibility in handling packed words, however, it is also necessary to provide the means for rearranging the positions of fields within words. This flexibility has been achieved in the word processor by the inclusion of five shift instructions in its repertorie. Two of these instructions shift the low-order 44 bits of a word, preserving sign bit positions 1 through 4. The remaining three instructions shift the entire 48 bits of the word.

The shift instructions cause the contents of the location specified by the A-address field to be placed in the accumulator, where they are shifted right the number of bit positions specified by the high-order six bits of the B-address field. The direction of the shift is always right-end-around, with the result that bits shifted out of the low-order position recirculate to the high-order position of the word. A 48-bit full word shift, therefore, produces the same result as a 0-bit shift. The effective number of bit positions by which a word may be shifted ranges from 0 to 44 for shifts protecting the sign of the operand, and from 0 to 48 for shifts of the entire word. If the number of bit positions specified to be shifted exceeds the maximum effective shift (it is possible to specify a 63-bit shift in the high-order six bits of B), the machine will actually perform the specified shift. The net number of places shifted as a result of the operation, however, will be the number of specified minus 44 in the case of shifts protecting sign or the number specified minus 48 in the case of shifts not protecting sign.

The shift instructions are inherent-mask instructions; that is, they are always performed

with a mask. The mask is applied to the shifted operand before delivery to the result location (specified by the C-address field). If the programmer wishes to deliver the entire word, therefore, he must specify a mask consisting of 48 binary ones. The location of the mask is designated by the low-order six bits of the B-address field, in conjunction with bits 2 through 9, 10 through 13, and 14 through 18 of the mask index register (see Figure 5-3, page 5-12). The two shift and substitute instructions and the Shift Word and Select instruction perform protected masking. This means that the values of the unmasked bit positions in the result location (those which do not correspond to binary ones in the mask) are preserved during the operation. The two shift and extract instructions perform unprotected masking; that is, the positions in the result location which do not correspond to binary ones in the mask are cleared to zeros. Protected masking is accomplished by gating the shifted contents of the accumulator and the original contents of the location specified by the C-address field into C under control of the mask in the mask register. Unprotected masking is accomplished by gating the contents of the accumulator and a generated word of all zeros into the location specified by the C-address field under control of the mask in the mask register.

Any type of addressing may be used in the A- and C-address fields of the shift instructions, with the exception that certain restrictions apply to the C address of a Shift Word and Select instruction. These will be discussed under the description of that instruction. If the B address of a shift instruction is all zeros, then no shifting is performed and the word at A is transferred to the location specified by C under control of a mask located by attaching six low-order zero bits to the partial address stored in the mask index register. If the B-address field is inactive, the behavior of the system is unspecified.

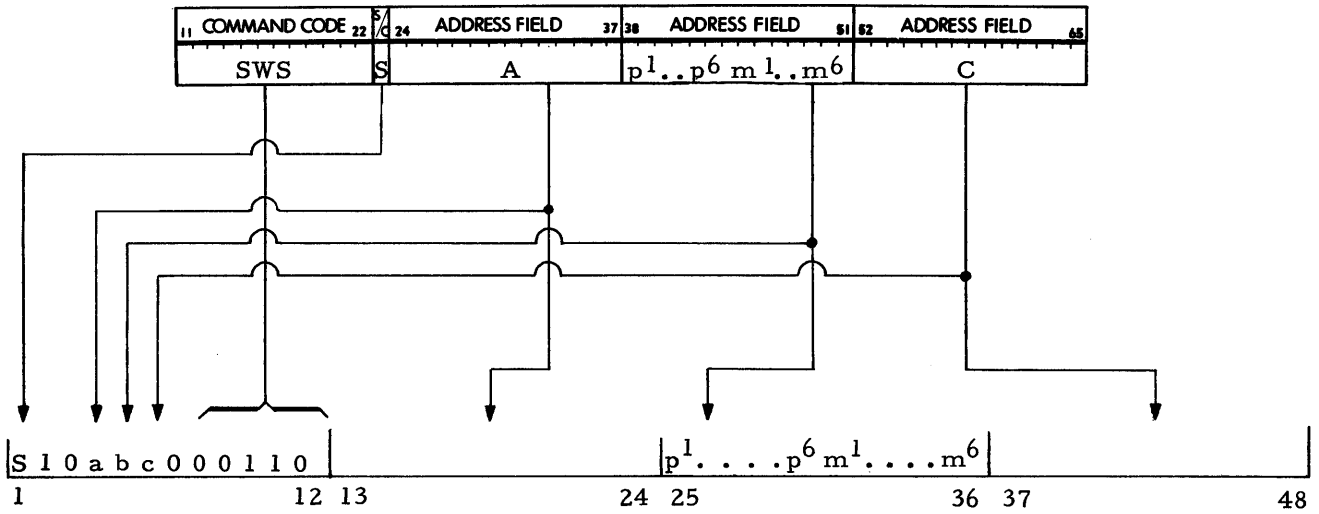
Mod 8 assembly notation for the shift instructions is designed to simplify the specification of masks and shifts. The symbolic tag of the mask is written in the command code field (rather than in the B-address field), separated from the operation code by a comma. The nature and extent of the shift is specified by three items of information in the B-address field, all separated by commas:

1. A character to designate the type of characters shifted: A = alphanumeric; D = hexadecimal; B or blank = binary.
2. A number to indicate the number of positions to be shifted (0 to 8, alphanumeric; 0 to 12, hexadecimal; 0 to 48, binary).
3. A character to designate the direction of the shift: L = left; R or blank = right.

Any valid assembly address format may be used in the A or C address of a shift instruction, subject only to the restrictions noted in connection with the C address of a Shift Word and Select instruction.

Assembly

All five of the instructions in the shift grouping are assembled as indicated in the following diagram (using the Shift Word and Substitute instruction):



SPS | SHIFT PRESERVING SIGN AND SUBSTITUTE

FORMAT

1 12 13 24 25 36 37 48

Command Code	Address Field	Address Field	Address Field
B10 abc 000 010 $(2002)_8$	A	$p^1 \ldots p^6 m^1 \ldots m^6$	C

FUNCTION

The Shift Preserving Sign and Substitute instruction directs the machine to shift the word at A, excluding sign bits 1 through 4, right-end-around the number of bit positions specified by the high-order six bits of the B-address field. Bits shifted out of position 48 recirculate to bit position 5. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register, as illustrated in Figure 5-3. The masked result is delivered to the location specified by C, protecting the bit positions of the contents of C which do not correspond to binary ones in the mask.

A shift of n bits to the left is specified in machine language by setting the high-order six bits of B equal to 44-n (in binary). Thus, a 12-bit shift to the left is effected by specifying a 32-bit shift (44 minus 12) to the right. In Mod 8 assembly language, the programmer may specify a left shift directly.

MASKING

Inherent mask.

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	All	p	All

Inactive:

The Shift Preserving Sign and Substitute instruction may use inactive addressing in the manner described in Table 5-19, on page 5- 83 .

SWS	SHIFT WORD AND SUBSTITUTE

FORMAT

1 12 13 24 25 36 37 48

Command Code	Address Field	Address Field	Address Field
B10 abc 000 110 $(2006)_8$	A	$p^1 \ldots p^6\ m^1 \ldots m^6$	C

FUNCTION

The Shift Word and Substitute instruction directs the machine to shift the entire word at A, including sign bits 1 through 4, right-end-around the number of bit positions specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register, as illustrated in Figure 5-3. The masked result is delivered to the location specified by C, protecting the bit positions of the contents of C which do not correspond to binary ones in the mask.

A shift of n bits to the left is specified in machine language by setting the high-order six bits of B equal to 48-n (in binary). A left shift may be specified directly in Mod 8 assembly language.

MASKING

Inherent mask.

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	All	p	All

Inactive:

The Shift Word and Substitute instruction may use inactive addressing in the manner described in Table 5-19, on page 5- 83.

SPE | SHIFT PRESERVING SIGN AND EXTRACT

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B10 abc 001 010 $(2012)_8$	A	$p^1 \ldots p^6 \; m^1 \ldots m^6$	C

FUNCTION

The Shift Preserving Sign and Extract instruction directs the machine to shift the word at A, excluding sign bits 1 through 4, right-end-around the number of bit positions specified by the high-order six bits of the B-address field. Bits shifted out of position 48 recirculate to bit position 5. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register, as illustrated in Figure 5-3. The masked result is delivered to the location specified by C, not protecting the bit positions which do not correspond to binary ones in the mask; it clears them to zeros when the result is stored.

A shift of n bits to the left is specified in machine language by setting the high-order six bits of B equal to 44-n (in binary). Thus, a 12-bit shift to the left is effected by specifying a 32-bit shift (44 minus 12) to the right. In Mod 8 assembly language the programmer may specify a shift directly.

MASKING

Inherent mask.

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	All	p	All

Inactive:

The Shift Preserving Sign and Extract instruction may use inactive addressing in the manner described in Table 5-20, on page 5- 84 .

SWE | SHIFT WORD AND EXTRACT

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B10 abc 001 110 $(2016)_8$	A	$p^1 \ldots p^6 \; m^1 \ldots m^6$	C

FUNCTION

The Shift Word and Extract instruction directs the machine to shift the entire word at A, including sign bits 1 through 4, right-end-around the number of bit positions specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register, as illustrated in Figure 5-3. The masked result is delivered to the location specified by C, not protecting the bit positions which do not correspond to binary ones in the mask; it clears them to zeros when the result is stored.

A shift of n bits to the left is specified in machine language by setting the high-order six bits of B equal to 48-n (in binary). A left shift may be specified directly in Mod 8 assembly language.

MASKING

Inherent mask.

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	All	p	All

Inactive:

The Shift Word and Extract instruction may use inactive addressing in the manner described in Table 5-20, on page 5- 84.

SSL | SHIFT WORD AND SELECT

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B10 abc 010 110 $(2026)_8$	A	$p^1 \ldots p^6\ m^1 \ldots m^6$	C

FUNCTION

The Shift Word and Select instruction causes the machine to shift the entire word at A right-end-around the number of bit positions specified by the high-order six bits of B. The result of the shift is masked, using a mask whose address is generated as in the Shift Preserving Sign and Substitute instruction, and this masked result, with a positive sign implied, is added in binary to the address specified in C to form a new address. The sequencing counter designated as the source of the next instruction is then changed so that the next instruction is selected from the location specified by the modified C-address field. Regardless of the mask used, the machine never adds more than 11 low-order bits to the C address.

The C address of a Shift Word and Select instruction may be a direct memory location address, an indirect memory location address, an indexed memory location address, or an indexed indirect memory location address in both the

normal and extended modes. For these four types of address, the masked result, called C', is used with the C address as described below:

Normal Mode

1. Direct Memory Location Address. C' is added to the low-order 11 bits of C, discarding the end carry, if any. The resulting 11 bits are placed in the specified sequencing counter, together with the sign, bank indicator, and array bits from the counter that selected the current instruction.

2. Indirect Memory Location Address. C' is added to the low-order 11 bits of C, discarding the end carry, if any. The result is used to select a control register whose contents, including the sign, bank indicator, and array bits, are transferred to the specified sequencing counter. The tabular bit of the modified C address is ignored. The contents of the addressed control register are incremented in the usual fashion after use, unless the control register addressed is the counter to be changed (either sequence or cosequence), in which case incrementing does not take place.

3. Indexed Memory Location Address. The contents of the specified index register are augmented in the usual fashion to form a complete 23-bit address. C' is then added to the low-order 11 bits of this complete address, discarding the end carry, if any. The resulting complete 23-bit address is transferred, with a positive sign, to the specified sequencing counter.

4. Indexed Indirect Memory Location Address. A complete control register address is generated by augmenting the contents of the specified index register in the usual fashion. C' is added to the low-order 11 bits of this generated address, discarding the end carry, if any. The result of this addition is used to select a control register whose contents, including the sign, bank indicator, and array bits are transferred to the specified sequencing counter. The tabular bit in the generated control register address is ignored. The contents of the control register selected are incremented in the usual fashion after use, unless this control register is the counter to be changed (either sequence or cosequence counter).

Extended Mode

1. Direct Memory Location Address. The low-order 11 bits (C') of the result of the masked shift are added to the address field (low-order 11 bits). The resulting 23 bits are placed in the specified sequencing counter, with a positive sign.

   NOTE: Carry-outs of bit 11 are not inhibited from being propagated.

2. Indirect Memory Location Address. This addressing type proceeds exactly as for the normal indirect main memory addressing with the exception that bits 12 through 23 of the address field are all zeros.

   NOTE: Carry-outs of bit 11 are not inhibited from being propagated.

3. Indexed Memory Location Address. The low-order 11 bits (C') of the result of the masked shift are added to the modified control register contents in the usual fashion to form a complete 23-bit address. The resulting complete 23-bit address is transferred, with a positive sign, to the specified sequencing counter.

   NOTE: Carry-outs of bit 11 are not inhibited from being propagated.

4. Indexed Indirect Memory Location Address. The low-order 11 bits (C') of the masked shift are added to the address field (low-order 11 bits). The resulting 23 bits are placed in the specified sequencing counter, with a positive sign.

   NOTE: Carry-outs of bit 11 are not inhibited from being propagated.

If a parity error is detected in a word selected from main memory during execution of a Shift Word and Select instruction, an unprogrammed transfer is made to U + 2 or U + 3, and the contents of the sequencing counter are not altered. However, the contents of the appropriate history register will be changed, and the bisequence bit in the program control register will be set in response to the bisequence bit in the command code of the instruction.

MASKING

Inherent mask.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	p	All

Inactive:

The Shift Word and Select instruction may use inactive addressing in the manner described in Table 5-21, on page 5- 85 .

Table 5-19. Unmasked Inactive Addressing for Shift Preserving Sign and Substitute (SPS) and Shift Word and Substitute (SWS) Instructions

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	When B is void, the behavior of the system is unspecified. [1]
INACTIVE	INACTIVE	ACTIVE	When B is void, the behavior of the system is unspecified.
INACTIVE	ACTIVE	INACTIVE	The contents of the accumulator are shifted right-end-around the number of bit positions specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register. [1]
INACTIVE	ACTIVE	ACTIVE	The contents of the accumulator are shifted right-end-around the number of bit positions specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register. The masked result is delivered to the location specified by C, protecting the bit positions of the contents of C which do not correspond to binary ones in the mask.

Table 5-19 (cont). Unmasked Inactive Addressing for Shift Preserving Sign and Substitute (SPS) and Shift Word and Substitute (SWS) Instructions

A	B	C	Action for void addressing
ACTIVE	INACTIVE	INACTIVE	When B is void, the behavior of the system is unspecified. [1]
ACTIVE	INACTIVE	ACTIVE	When B is void, the behavior of the system is unspecified.
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by A are shifted right-end-around the number of bit positions specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register. The masked result is placed in the accumulator. [1]
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by A are shifted right-end-around the number of bit positions specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register. The masked result is delivered to the location specified by C, protecting the bit positions of the contents of C which do not correspond to binary ones in the mask.

[1] Hunting is inhibited.

Table 5-20. Unmasked Inactive Addressing for Shift Preserving Sign and Extract (SPE) and Shift Word and Extract (SWE) Instructions

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	When B is void, the behavior of the system is unspecified. [1]
INACTIVE	INACTIVE	ACTIVE	When B is void, the behavior of the system is unspecified. [1]
INACTIVE	ACTIVE	INACTIVE	The contents of the accumulator are shifted right-end-around the number of bit positions specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register. [1]
INACTIVE	ACTIVE	ACTIVE	The contents of the accumulator are shifted right-end-around the number of bit positions specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register. The masked

Table 5-20 (cont). Unmasked Inactive Addressing for Shift Preserving Sign and Extract (SPE) and Shift Word and Extract (SWE) Instructions

A	B	C	Action for void addressing
INACTIVE	ACTIVE	ACTIVE	result is delivered to the location specified by C, not protecting the bit positions which do not correspond to binary ones in the mask; it clears them to zeros when the result is stored.
ACTIVE	INACTIVE	INACTIVE	When B is void, the behavior of the system is unspecified.[1]
ACTIVE	INACTIVE	ACTIVE	When B is void, the behavior of the system is unspecified.
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by A are shifted right-end-around the number of bit positions specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register. The masked result is placed in the accumulator.[1]
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by A are shifted right-end-around the number of bit positions specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register. The masked result is delivered to the location specified by C, not protecting the bit positions which do not correspond to binary ones in the mask; it clears them to zeros when the result is stored.

[1] Hunting is inhibited.

Table 5-21. Unmasked Inactive Addressing for Shift Word and Select (SSL) Instruction

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	When B is void, the behavior of the system is unspecified.[1]
INACTIVE	INACTIVE	ACTIVE	When B is void, the behavior of the system is unspecified.
INACTIVE	ACTIVE	INACTIVE	The contents of the accumulator are shifted right-end-around the number of bit positions specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register.[1]
INACTIVE	ACTIVE	ACTIVE	The contents of the accumulator are shifted right-end-around the number of bit positions

Table 5-21 (cont). Unmasked Inactive Addressing for Shift Word and Select (SSL) Instruction

A	B	C	Action for void addressing
INACTIVE	ACTIVE	ACTIVE	specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register. The masked result, with a positive sign implied, is added in binary to the address specified in C to form a new address. The sequencing counter designated as the source of the next instruction is then changed so that the next instruction is selected from the location specified by the modified C-address field.
ACTIVE	INACTIVE	INACTIVE	When B is void, the behavior of the system is unspecified. [1]
ACTIVE	INACTIVE	ACTIVE	When B is void, the behavior of the system is unspecified.
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by A are shifted right-end-around the number of bit positions specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask register. The masked result is placed in the accumulator. [1]
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by A are shifted right-end-around the number of bit positions specified by the high-order six bits of the B-address field. The result is masked using a mask whose address is generated from the low-order six bits of B and the partial address stored in the mask index register. The masked result, with a positive sign implied, is added in binary to the address specified in C to form a new address. The sequencing counter designated as the source of the next instruction is then changed so that the next instruction is selected from the location specified by the modified C-address field.

[1] Hunting is inhibited.

MISCELLANEOUS INSTRUCTIONS

Control Program (MPC), Proceed (PR), and Simulator (S), unlike all other instructions in the word processor's repertorie, belong to no particular grouping of instructions. The operation of each is unique and varies from all others.

MPC	CONTROL PROGRAM

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B10 abc 000 000 $(2000)_8$	(A) not used	B	C

FUNCTION

To facilitate multiprogramming the word processor contains a nonaddressable 48-bit register which is used to store information required for efficient control of a multiprogram operation. Access to this register, called the program control register (PCR), is provided by the machine instruction Control Program (MPC). The information stored in the program control register is described below:

Bits 1-16: Reserved for master control use.

Bit 17: Master Control Indicator. A one in this position indicates that master control selected the current instruction. A zero indicates that one of the regular eight groups of the word processor selected the current instruction.

Bits 18-20: Control Group Indicators. These bits, when bit 17 is a zero, denote the control register group under whose control the MPC instruction was selected. When bit 17 is a one, indicating master control is in charge, bits 2 through 4 indicate a master-control-stored Interrupt status. When bit 17 is a one, a one in bit 19 designates the master control sequence mode and a zero designates the cosequence mode. Also, a one in bit 20 indicates master control hunt mode and a zero indicates the no-hunt mode (this bit is derived from the B memory designator bit of a master control MPC instruction).

Bits 21-28: Bisequence Bits (one per regular control register group). Each such bit is a one if the next instruction to be executed under control of the respective control register group is to be selected by the cosequence counter or a zero if the next such instruction is to be selected by the sequence counter.

Bits 29-36: Program Demand Bits (one per control register group). Each such bit is a one if the corresponding sequence counter or cosequence counter is in demand and a zero otherwise. By "in demand" is meant program running, including the case in which the instruction was trapped to master control, e.g., certain peripheral orders.

Bits 37-40: Console Fixed-Start Bits. The values of these four bits are generated when a hexadecimal key is struck as part of a console fixed-start instruction. The operator uses this instruction to restart a program without knowing the location from which the start is to be made; he specifies

Bits 37-40: (cont) the group indicator of the controlling control register group, followed by a hexadecimal digit which the programmer specifies. The console fixed-start bits retain the generated values until another console fixed-start is executed.

Bits 41-48: Reserved for master control use.

For any group of eight bits which represent the eight control register groups (bisequence bits, program demand bits), the highest-order bit corresponds to control register group 0, and the succeeding bits correspond to groups 1, 2, 3, etc. For example, bit 29 is the program demand bit for group 0, bit 30 represents group 1 and so forth up to bit 36, which represents group 7. Figure 5-23 summarizes the control register group indicator relationships.

The Control Program instruction is defined as follows. Ignore the A-address field. Place the contents of the program control register in the location specified by the C-address field. Then, alter the bits of the program control register specified by bits 5 through 12 of the B-address field of this instruction. Bits 1 through 4 of the B address define the manner in which the bits of the program control register are altered. Hunt for the next program in demand if the memory designator for the B address (bit 5 of the command code) is one; otherwise, do not hunt. Table 5-22 shows the manner in which B_{1-4} (the high-order four bits of B) determine the interpretation of B_{5-12} (the low-order eight bits of B).

The command code for the Control Program instruction allows the programmer to specify whether the next instruction shall be selected by the sequence or cosequence counter. However, if a conflict exists between the value assigned to a bisequence bit by the B address of a Control Program instruction and the value assigned to the same bit by command code bit 1 of the same instruction, the value corresponding to bit 1 takes precedence. The affected bisequence bit in the PCR is set according to the value of bit 1, but not until the PCR contents have been stored.

Since the A address of the instruction is ignored, it may be used to store 12 bits of data at the discretion of the programmer. The A address designator bit may be zero or one. The C address may be a direct, indexed, or indirect memory location address.

Control Register Group Indicator (PCR Bits 18-20)	Program Demand Bits PCR Position	Bisequence Bits PCR Position	Peripheral Channel Address (Command Code Bits 1-3)
0	29	21	0
1	30	22	1
2	31	23	2
3	32	24	3
4	33	25	4
5	34	26	5
6	35	27	6
7	36	28	7

Figure 5-23. Word Processor Control Register Group Indicator Relationships

Table 5-22. B-Address Field Function in Control Program (MPC) Instruction

	B_{1-4}	Mnemonic Code Where Applicable	Interpretation of B_{5-12}*
These five combinations do not affect the bisequence bits.	0000	STOP	Turn off the program which initiated this instruction
	0001	DOFF	Turn off all program demand bits corresponding to zeros in B_{5-12}; do not alter program demand bits corresponding to ones.
	0010	DON	Turn on all program demand bits corresponding to ones in B_{5-12}; do not alter program demand bits corresponding to zeros.
	0011	MPC	Turn on all program demand bits corresponding to ones in B_{5-12}; turn off all program demand bits corresponding to zeros.
	1001	MON	Do not alter the contents of the program control register. Initiates a call to master control.
These five combinations may affect the bisequence bits.	0100	SPCR	Do not alter the contents of the program control register.
	0101	SCON	Turn on the program demand bits and set to SC the bisequence bits corresponding to zeros in B_{5-12}; do not alter the program demand bits or the bisequence bits corresponding to ones in B_{5-12}.
	0110	CSCON	Turn on the program demand bits and set to CSC the bisequence bits corresponding to ones in B_{5-12}; do not alter the program demand bits or the bisequence bits corresponding to zeros in B_{5-12}.
	0111	MPC	Do not alter any program demand bits; set to SC the bisequence bits corresponding to zeros in B_{5-12}; set to CSC the bisequence bits corresponding to ones in B_{5-12}.
	1000	MPC	Turn on the program demand bit and set to SC the bisequence bit corresponding to a one in B_5 or B_7. If B_5 and B_7 are both ones, the behavior of the system is unspecified. Select the next instruction from the group thus turned on. Turn off the group which selected this instruction (but do not alter the bisequence bit) if the corresponding bit (B_6 or B_{8-12}) is a zero. If the corresponding bit is a one, do not alter the program demand bit or the bisequence bit for this group. Do not alter the program demand bit or bisequence bit of any other program. If the program demand bit corresponding to a one in B_5 or B_7 is already on, do not execute this instruction but stall the group which selected it until the specified program demand bit is turned off.

*Bits 5 through 12 of the B address are in one-to-one correspondence with the bisequence bits (21 through 28) and the program demand bits (29 through 36) of the program control register. In other words, B_5 controls control register group 0, B_6 controls control register group 1, etc.

NOTES: Only the groups assigned to the job executing the MPC instruction are affected. If all groups of a job are "off," the program will be terminated.

From the standpoint of hunting for the next program in demand, it is irrelevant for this one instruction whether the C address is active or inactive. Hunting is controlled by the B-address designator bit in the command code; a one in this bit allows hunting. In addition, hunting always occurs if the program executing the Control Program instruction is stalled (see B_{1-4} = 1000, Table 5-22). If the C address is inactive, the contents of the program control register are stored in both the accumulator and the low-order product register. Following a Control Program instruction with inactive C address which does not allow hunting (indicated by 0 in the B-designator bit), the former contents of the program control register may be retrieved from the accumulator by immediately executing a Word Add (WA) instruction with inactive A and B addresses, or from the low-order product register by immediately executing a Transfer and Sequence Change (TS) instruction with an inactive A address.

Multiprogram control operates in such a way that the next active control group is selected during the execution of an instruction that allows hunting. For this reason, execution in the hunt mode of a Control Program instruction which turns on the demand for a new group without a group already in demand intervening will result in one complete search cycle before the execution of an instruction in the newly activated group. On the other hand, execution in the hunt mode of a Control Program instruction which turns off the demand of the next selected group will cause multiprogram control to resume searching for the next active group.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	Not Used	p	D, I, H, IH

Inactive:

The Control Program instruction may use inactive addressing in the manner described in Table 5-23, on page 5-93.

PR | PROCEED

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
X00 0XX 000 000 $(0000)_8$	(A) irrelevant	(B) irrelevant	(C) irrelevant or 111 111 111 111

FUNCTION

The Proceed instruction results in no operation other than the normal incrementing of the sequencing counter that selected it. Bits 1, 5, and 6 of the command code are irrelevant, while bit 4 must be a zero; therefore, the Proceed instruction cannot specify the source of the next instruction and is always followed by an instruction selected by the same sequencing counter.

Because the A-, B-, and C-address fields are irrelevant, they can be used to store information. However, if the information stored in the C-address field consists of 12 binary ones (inactive address), hunting for the next program in demand is inhibited.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

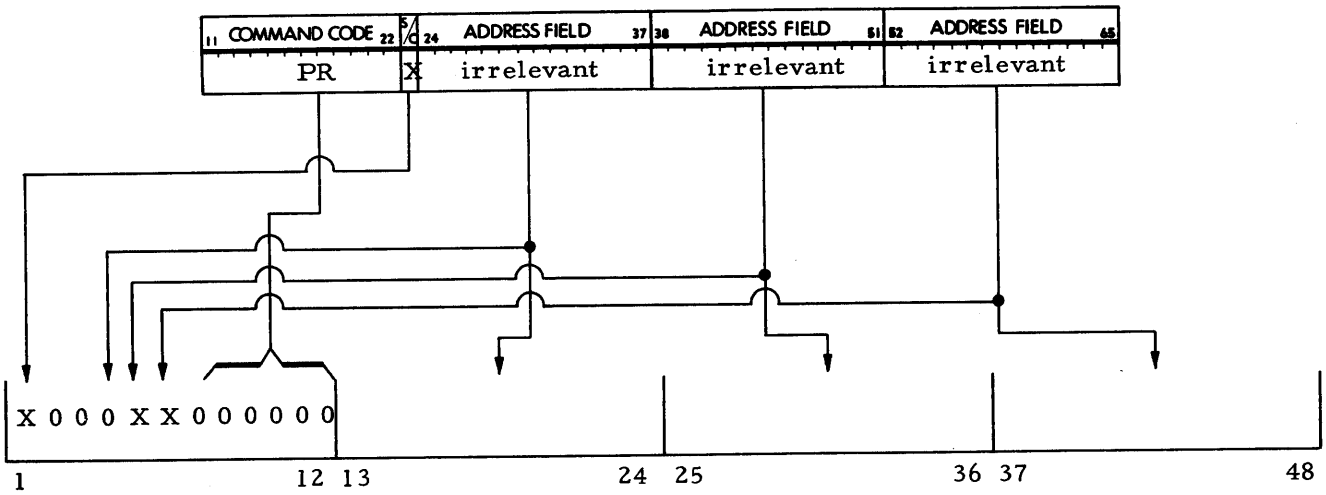
Active:	A	B	C
Unmasked	irrelevant	irrelevant	irrelevant

Inactive:

Inactive addressing for this instruction is not detected and has no effect.

Assembly

This instruction is assembled as indicated in the following diagram:



S SIMULATOR

FORMAT

1 12 13 24 25 36 37 48

Command Code	Address Field	Address Field	Address Field
0XX XXX XXX 111 (Direct) 1XX XXX XXX 111 (Indexed) $(0007)_8$	A	(B) not used	C

FUNCTION

Simulator instructions permit the programmer to represent a subroutine with a single instruction in his program. For each Simulator instruction used, he codes a subroutine which is stored elsewhere in memory, beginning with the next memory location higher than the address specified by the command code. The Simulator instruction sets the cosequence counter to the starting address of the subroutine and then gives control to this counter.

A Simulator instruction is specified by a machine command code in which the low-order three bits are ones. If the high-order bit of the command code is a zero, the low-order 11 bits are interpreted as a main memory subaddress ending in octal seven. If the high-order bit is a one, the low-order 11 bits are interpreted as a 3-bit index register number and an 8-bit augmenter ending in octal seven. Since the high-order bit of the command code is used to indicate the type of addressing, the programmer does not have the option of specifying the source of the next instruction. In fact, the next instruction is automatically taken from the cosequence counter. This is the only instance in which the machine makes a functional distinction between the sequence counter and the cosequence counter.

When a Simulator instruction is executed, the instruction itself is stored in the location specified by the command code. If this address refers directly to a memory location, then the instruction is stored in the same bank of memory from which it was executed; if the address is indexed, the instruction may be stored in any bank, according to the value of the bank indicator bits in the specified index register. The cosequence counter is set to the next higher address and the next instruction is taken from the cosequence counter. The contents of the source counter, after normal incrementing, are stored in the cosequence history register to provide a return to the main program.

Since the address portions of a Simulator instruction have no assigned functions, they may be used to store subroutine parameters such as the sources of operands, the location in which a result is to be stored, the number of words to be manipulated, and so forth. Since the machine automatically stores in control register AU1 an address generated from the A-address field and in AU2 an address generated from the C-address field, it is advantageous to use these address fields for operand or result locations. The A- and C-address fields may contain direct memory location addresses (bit 1 = zero) or indexed memory location addresses (bit 1 = one). If bit 1 is zero, the low-order 11 bits of the address field, the 8-bit array, and the 4-bit bank indicator from the sequencing counter which selected the instruction are placed in the arithmetic control counter with a positive sign to form a complete address. If bit 1 is one, the augmented contents of the specified index register are placed in the counter with a positive sign. The contents of the locations whose addresses are stored in the two counters may then be addressed easily through the technique of indirect memory location addressing.

The command code for a Simulator instruction is S, followed by a comma and an address designated by a symbolic tag or by an index register number with augmenter equal to seven. The A- and C-addres fields may contain symbolic tags, with or without address modifiers, or index register numbers with augmenters.

NOTE: The Simulator instruction is not valid in the extended mode.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

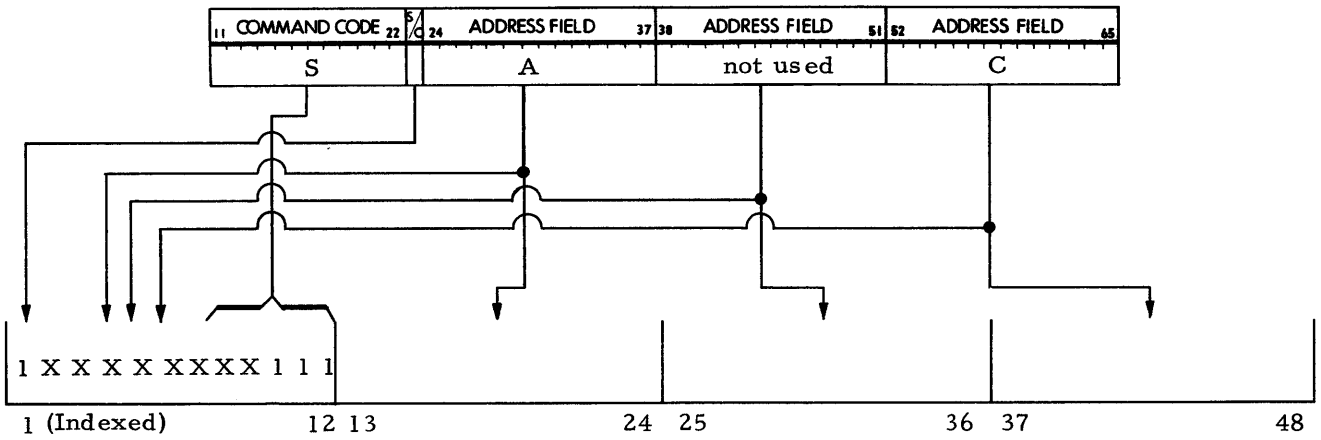
Active:	A	B	C
Unmasked	D, I	not used	D, I

Inactive:

The Simulator instruction may use inactive addressing in the manner described in Table 5-24, on page 5-94.

Assembly

This instruction is assembled as indicated in the following diagram:



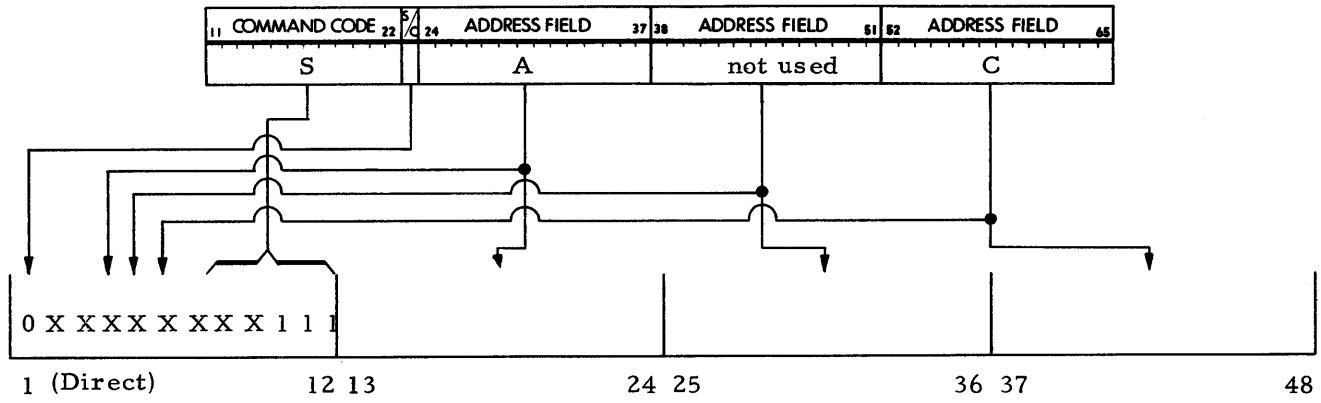


Table 5-23. Unmasked Inactive Addressing for Control Program (MPC) Instruction

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	The instruction is not performed, resulting in a call to master control.
INACTIVE	INACTIVE	ACTIVE	The instruction is not performed, resulting in a call to master control.
INACTIVE	ACTIVE	INACTIVE	The contents of the program control register are delivered to both the accumulator and the

Table 5-23 (cont). Unmasked Inactive Addressing for Control Program (MPC) Instruction

A	B	C	Action for void addressing
INACTIVE	ACTIVE	INACTIVE	low-order product register. A call to master control will then occur, resulting in master control performing alter operations (see Table 5-20).
INACTIVE	ACTIVE	ACTIVE	The contents of the program control register are delivered to both the accumulator and the low-order product register. A call to master control will then occur, resulting in master control performing alter operations (see Table 5-20).
ACTIVE	INACTIVE	INACTIVE	The instruction is not performed, resulting in a call to master control.
ACTIVE	INACTIVE	ACTIVE	The instruction is not performed, resulting in a call to master control.
ACTIVE	ACTIVE	INACTIVE	The contents of the program control register are delivered to both the accumulator and the low-order product register. A call to master control will then occur, resulting in master control performing alter operations (see Table 5-20).
ACTIVE	ACTIVE	ACTIVE	The contents of the program control register are delivered to both the accumulator and the low-order product register. A call to master control will then occur, resulting in master control performing alter operations (see Table 5-20).

Table 5-24. Unmasked Inactive Addressing for the Simulator (S) Instruction

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	No operation takes place. [1]
INACTIVE	INACTIVE	ACTIVE	The address generated from the location specified by C is placed in control register AU2.
INACTIVE	ACTIVE	INACTIVE	No operation takes place. [1]
INACTIVE	ACTIVE	ACTIVE	The address generated from the location specified by C is placed in control register AU2.
ACTIVE	INACTIVE	INACTIVE	The address generated from the location specified by A is placed in the control register AU1. [1]
ACTIVE	INACTIVE	ACTIVE	The address generated from the location specified by A is placed in the control register AU1; the address generated by the location specified by C is placed in control register AU2.
ACTIVE	ACTIVE	INACTIVE	The address generated from the location specified by A is placed in the control register AU1. [1]
ACTIVE	ACTIVE	ACTIVE	The address generated from the location specified by A is placed in the control register AU1; the address generated by the location specified by B is placed in control register AU2.

[1] Hunting is inhibited.

LOGICAL INSTRUCTIONS

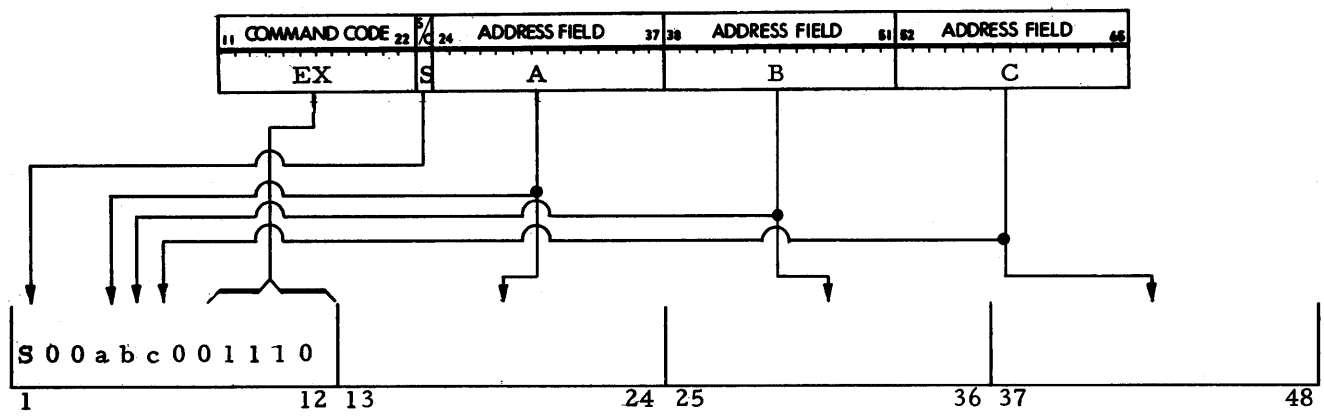
The four instructions which make up the logical group manipulate words on an individual-bit basis, combining bits from two words to form a third. The rules by which the bits are combined are similar to the rules under which the logical elements of a computer operate; hence the name logical instructions. All operands are regarded as 48-bit words in which each bit is an individual unit of information unrelated to any other bit.

The four logical instructions are Extract, Substitute, Half Add, and Superimpose. Extract and Substitute have command codes of the "inherent mask" format and utilize the entire B-address field to specify the location of a mask. Half Add and Superimpose have command codes of the "general, masked or unmasked" format.

The logical instructions may address control registers in one or more address field. The result of such an operation may be determined by applying the rules governing the transfer of a control register word to a 48-bit register and the transfer of a 48-bit word to a control register.

Assembly

All the instructions in the logical grouping are assembled as indicated in the following diagram (using the Extract instruction):



EX	EXTRACT

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B00 abc 001 110 $(0016)_8$	A	B	C

FUNCTION

The Extract instruction places the A operand in the location specified by the C-address field, using the B operand as a mask and not protecting the unmasked portions of C. (The mask index register is not used in locating the mask.) This is equivalent to combining the corresponding bits of the A and B operands in accordance with the following rule:

> If corresponding bit positions in the word at A and the word at B both contain ones, the result shall contain a one in this position. In all other cases, the result shall contain a zero. This is the "logical AND" function.

The use of inactive addressing with the Extract instruction provides access to the 48-bit arithmetic register call the mask register. When a mask is specified in an instruction, either in the command code or in the B-address field, the contents of the specified location are placed in the mask register where they remain until a subsequent instruction calls for a mask. Therefore, the contents of the mask register are unrelated to the instruction being performed if this instruction does not specify a mask. Since the mask register has no address, the programmer cannot transfer its contents directly to memory. However, by executing an Extract instruction which specified the address of a word of 48 binary ones in the A-address field and an inactive address in B, the programmer can store in the location specified by C a word guaranteed to be identical to the contents of the mask register. If the programmer wishes to insert a full word into the mask register without disturbing any memory location, he may perform an Extract instruction with an inactive C address.

MASKING

Inherent mask.

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	All	All	All

Inactive:

> The Extract instruction may use inactive addressing in the manner described in Table 5-25, on page 5-99.

SS | SUBSTITUTE

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B00 abc 000 110 $(0006)_8$	A	B	C

FUNCTION

The Substitute instruction performs the same general function as the Extract instruction except that the contents of the location specified by the C-address field are protected. When this instruction is executed, therefore, the computer places the 48-bit contents of A in the accumulator and the 48-bit contents of B in the mask register and then forms a new word according to the following rule:

> Whenever the B operand contains a one bit, the result contains the corresponding bit of the A operand. Whenever the B operand contains a zero bit, the result contains the corresponding bit of the word at C.

Finally, the result is stored in the location specified by the C-address field. The result of the operation may differ from the result of an Extract instruction having the same operands only in those bit positions for which the B operand has a value of zero.

MASKING

Inherent mask.

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	All	All	All

Inactive:

> The Substitute instruction may use inactive addressing in the manner described in Table 5-25, on page 5-99.

HA | HALF ADD

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B10 abc 010 101 $(2025)_8$	A	B	C

FUNCTION

The Half Add instruction performs a binary addition of the 48-bit A and B operands in the accumulator, discarding all carries, and stores the result in the location specified by the C-address field. This is equivalent to combining the corresponding bits of the A and B operands in accordance with the following rule:

> If the corresponding bit positions in the A and B operands have the same value, the result shall contain a zero in this position. In all other cases, the result shall contain a one. This is the "logical exclusive OR" function.

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 110 101 $(0065)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D, I	D, I	D, I
Unmasked	All	All	All

Inactive:

Both masked and unmasked versions of the Half Add instruction may use inactive addressing in the manner described in Tables 5-4 and 5-5, on pages 5-51 and 5-52.

SM | SUPERIMPOSE

FORMAT

1 12	13 24	25 36	37 48
Command Code	Address Field	Address Field	Address Field
B10 abc 000 101 $(2005)_8$	A	B	C

FUNCTION

The Superimpose instruction performs a superposition of the 48-bit A and B operands in such a way that the result contains a one in every position in which a one existed in either A or B or both, and stores the result in the location specified by the C-address field. This is equivalent to combining the corresponding bits of the A and B operands in accordance with the following rule:

If corresponding bit positions of the A and B operands are both zero, the result shall contain a zero in this position. In all other cases, the result shall contain a one. This is the "logical inclusive OR" function.

MASKING

This instruction may be written in either masked or unmasked form. The masked command code appears as follows:

BMM MMM 100 101 $(0045)_8$

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	D, I	D, I	D, I
Unmasked	All	All	All

Inactive:

Both masked and unmasked versions of the Superimpose instruction may use inactive addressing in the manner described in Tables 5-4 and 5-5, on pages 5-51 and 5-52.

Table 5-25. Unmasked Inactive Addressing for the Extract (EX) and Substitute (SS) Instructions[1]

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	No operation takes place.[2]
INACTIVE	INACTIVE	ACTIVE	The contents of the accumulator are placed in the location specified by C using the contents of the mask register as a mask, and not protecting the unmasked portions of C.
INACTIVE	ACTIVE	INACTIVE	The contents of the accumulator are placed into the mask register.[2]
INACTIVE	ACTIVE	ACTIVE	The contents of the accumulator are placed in the location specified by C using the B operand as a mask, and not protecting the unmasked portions of C.
ACTIVE	INACTIVE	INACTIVE	The contents of the location specified by A are placed in the mask register.[2]
ACTIVE	INACTIVE	ACTIVE	The contents of the location specified by A are placed in the location specified by C using the contents of the mask register as a mask, and not protecting the unmasked portions of C.
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by A are placed into the mask register.[2]
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by A are placed in the location specified by C, using the B operand as a mask, and not protecting the unmasked portions of C.

[1]The unmasked inactive addressing for the Substitute instruction is the same as for the Extract instruction (above), with the exception that the unmasked portions of a C address are protected.

[2]Hunting is inhibited.

INPUT/OUTPUT CONTROL OPERATIONS

Effective control over data transfers between the central processor and peripheral units and over the peripheral units themselves is maintained by the use of two basic instructions: Peripheral Control and Transfer (PCT), and Peripheral Control and Branch (PCB). The PCB instruction performs two distinct functions: (1) it initiates strictly mechanical operations such as magnetic tape rewinds and card rejections; and (2) it causes a program branch to be performed contingent upon the setting of peripheral condition indicators. The latter facility allows programmed tests for such peripheral conditions as read or write errors, busy peripheral devices

or controls and magnetic tape unit at end of tape. In addition to being able to perform the same test and control functions as the PCB instruction, the PCT instruction is used to initiate data transfer operations and certain other related operations, such as backspace magnetic tape and advance the printer form.

PCB | PERIPHERAL CONTROL AND BRANCH

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
001 000 000 000 $(1000)_8$	inactive	B (see below)	C

FUNCTION

The Peripheral Control and Branch (PCB) order can initiate test and control operations, in a manner similar to the PCB instruction in the VLF (variable-length-field) processor.

The C address specifies a main memory location to which the program branches if a negative response is made to the test condition or control operations specified in the stream of control characters. A positive response indicates either a "no branch" answer to the test conditions, or that an operation has been successfully initiated. For a "no branch" condition, the next order is taken from the controlling counter. The order does not hunt if it branches.

The main memory location (and subsequent location) specified by the B address contain the stream of control characters for the I/O controller. These characters specify the test and control operations to be performed. The 48 bits of each of these locations are considered to be eight 6-bit characters; the function of each character is identical to the control characters in a PCB instruction of the VLF processor (see Section VI). If more than 8 characters are needed, C7 to C14 will be the contents of the location specified by the address B + 1, etc., to the number required.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	Inactive	D, I	D, I

PCT	PERIPHERAL CONTROL AND TRANSFER

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
$001C_1C_2C_3, 000000$ $(1C00)_8$	A	B (see below)	C

FUNCTION

The Peripheral Control and Transfer (PCT) instruction can (1) initiate both test and control operations and (2) transfer data between a peripheral device and main memory.

The C address specifies a main memory location to which the program branches if a negative response is made to the test conditions or control operations specified in the stream of control characters in the B field (data transfer is not initiated). A positive response indicates a "no branch" answer to test conditions and successful initiation of an operation. If a "no branch" condition occurs, the next order is taken from the controlling counter. The order does not hunt if it branches.

The contents of the location specified by address B are identical to the B address in a PCB instruction. The contents of the location with address B + 1 contain information to initiate a data transfer between main memory (starting location is the A address; character positions are indicated by bits 4 through 6 (C_1, C_2, C_3) of the command code field) and the same peripheral device (using the same counter and time slot) to which the test and control operations were directed. This data transfer will be performed only if the control and test operations resulted in a positive response (no branch). If bit 2 of the command code is a zero, the data transfer will terminate according to the data medium employed, i.e., punctuation. If bit 2 is a one, the data transfer will either terminate on an end-of-record (EOR) word[1] or be terminated by the input device.

The significance of the characters specified in the location with address B + 1, and subsequent locations, will be identical to the PDT instruction of the VLF processor.

MASKING

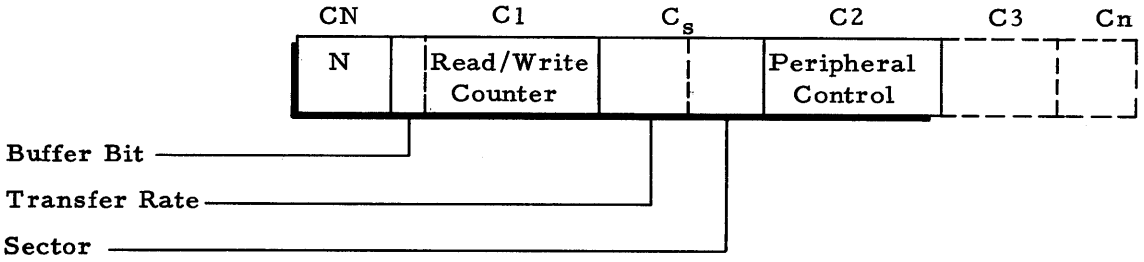
This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	D, I	D, I	D, I

[1] For an explanation of how EOR words are used to terminate data transfer, refer to the Models 800 or 1800 Programmers' Reference Manuals.

CONTROL CHARACTER SEQUENCE FOR THE PERIPHERAL CONTROL AND BRANCH AND PERIPHERAL CONTROL AND TRANSFER INSTRUCTIONS



CN Code

Encoded with a binary number (000000 - 111111) which is the total number of control characters in the sequence, not including CN or C1.

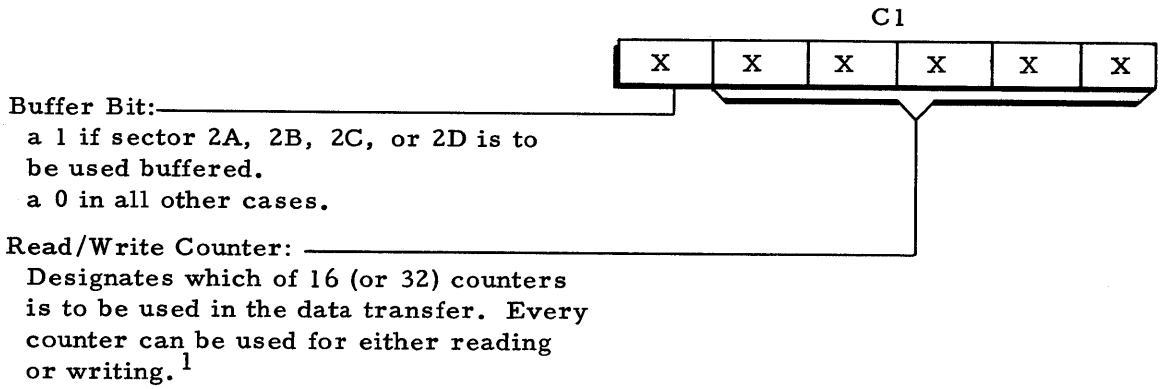
If CN = 000000, a PCB instruction will branch, if the read/write counter designated C1 is busy. A PCT instruction will branch unconditionally.

If CN = 000001, a PCB will branch as with a CN code of 000000. The PCB will also branch if sufficient time slots are not available on the sector designated by control character C_s. A PCT will branch unconditionally.

If CN = 000010, a PCB will branch as with a CN code of 000001; the C2 character is ignored. A PCT will branch as with a CN code of 000001, and will also branch if the peripheral control is busy.

If CN = 000011 or greater, a PCB or PCT will branch as with a CN code of 000001. The PCB or PCT will also branch if any test made by C3, etc., on the peripheral control designated by C2 is satisfied.

Read/Write Counter (C1)



Cs Code

This character specifies the data transmission rate and the sector to be used. The left and right halves are encoded separately as follows:

[1] In the basic Model 8200, the read/write code is 000000 to 011111; in the expanded Model 8200 (with Feature 8214), the read/write code is 000000 to 111111.

DATA TRANSFER RATE (thousands of characters per second)	Cs (Octal)	
	RATE	SECTOR
0	0	
83*	1	
167	2	
250	3	
333	4	
not legal	5	
500	6	
not legal	7	

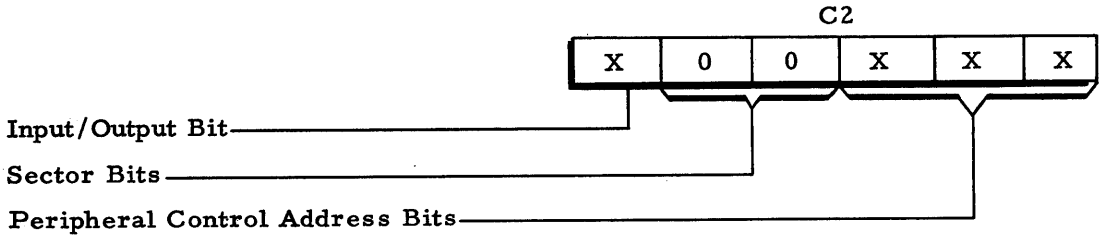
	SECTOR
0	Sector 1
1	not legal
2	Sector 2**
3	Sector 3
4	Sector 2A
5	Sector 2B
6	Sector 2C
7	Sector 2D

*Divide by 4 if used with sectors 2A through 2D, unbuffered.
**If Feature 8214 is installed, the sector digit 2 will be interpreted as specifying sector 2A in buffered mode (regardless of the buffer bit in the read/write character).

If CN is greater than or equal to 000011, and the rate is specified as 0, a PCB will test neither the read/write counter nor sector, but will branch (or not branch) as determined by C2, C3, etc., only.

Peripheral Control Designation (C2)

This 6-bit character specifies the logical address of the peripheral control to be used in the data transfer.



Input/Output Bit: This bit specifies the direction of data transfer when a peripheral control capable of both reading and writing is involved in the transfer. When such a bidirectional control is used,

0 = transfer data from memory to the peripheral control (output),
1 = transfer data to memory from the peripheral control (input).

Sector Bits: Sector bits must be zeros in Model 8200 peripheral addresses.

Peripheral Control Address Bits: These three bits, in conjunction with the preceding three bits, identify the address of the peripheral control involved in the operation.

C3 Through C_n (for Peripheral Control and Branch Instruction)

The specific use of these control characters is dependent upon the type of peripheral device addressed. A summary of coding for these characters (similar to those on the VLF processor) may be found in Appendix C.

C3 Through C6 (for Peripheral Control and Transfer Instruction)

The coding of these characters is the same as for the corresponding characters in the PCB instruction of the VLF processor. A summary of coding for these characters may be found in Appendix C.

C7 Through C_n (for Peripheral Control and Transfer Instruction)

These characters control data transfer as noted under the Peripheral Data Transfer (PDT) instruction in the VLF processor; characters C3 through C_n of a PDT instruction are identical to the contents for characters C7 through C_n of a PCT instruction. A summary for these characters may be found in Appendix C.

Additional Instructions

For the purpose of compatibility with 3/4-inch magnetic tape units of the Models 800/1800, the Model 8200 will handle the following instructions as part of its repertoire: Write Forward (WF), Read Forward (RF), Read Backward (RB), Rewind (RW), Compute Orthocount (CC), and Check Parity (CP), In addition, the 8200 can also perform the console Print instructions (PRA, PRO, PRD).

A detailed description of these instructions may be found in either the Model 800 Programmers' Reference Manual, Order Number 064, or the Model 1800 Programmers' Reference Manual, Order Number 053.

SECTION VI
VLF PROCESSING SUBSYSTEM

VLF PROCESSOR

The VLF processing subsystem, like the word processing subsystem, consists of a control unit, control memory, and an arithmetic unit.

The control unit is the hub of VLF processor activities. Its major function is to select, interpret, and execute all of the instructions in the stored program. In executing these instructions, the control unit coordinates the various activities of receiving data from and sending data to the I/O controller and transferring data within the VLF processor.

The control memory is a solid-state storage unit consisting of individually addressable control registers. Normally, control registers contain the addresses of instructions and of the data being processed during a program run. The storage unit provides storage for both the program instructions and the data to be processed according to these instructions.

The control registers of the VLF processor can be addressed either by programmed instruction or from the operator's console. For instance, an instruction can change the course of a program by manipulating the contents of the control register that governs program sequence; the operator can interrogate a control register to determine the exact location at which the program is halted, etc. When a register is addressed by programmed instruction, it is specified by means of a variant character in the instruction. A register is addressed from the console by using the register's octal address. The functional name of each register is listed in Table 6-1.

Table 6-1. VLF Processor Control Registers

NUMBER	FUNCTION
1	Sequence Register
1	Change Sequence Register
1	A-Address Register
1	B-Address Register
32	Input/Output Control Registers
1	Internal Interrupt Register
1	External Interrupt Register
7	Work Registers[1]

[1] These registers are available only to the VLF processor and must not be addressed by the program.

In addition to the above listed control registers, the VLF processor is equipped with four special purpose registers: an index barricade register, an internal base relocation register, a base relocation register controlled by the master control facility, and a key register.

The internal base relocation register, together with the index barricade register, provides a relocatable, two-fence, storage protection system within the total memory area assigned to the VLF processor. The base relocation register contains the starting address of the total memory area assigned to the VLF processor. The key register contains the key for this area of memory.

The arithmetic unit performs arithmetical and logical operations. It is composed of an adder (capable of performing both binary and decimal arithmetic, including decimal multiply and divide) and two operand storage registers.[1] The adder and operand storage registers can store four characters at a time. Result conditions such as overflow and zero balance cause the setting of indicators whose status may be used to initiate a change in program sequence. In general, an arithmetical logical operation is performed as follows.

1. An instruction in the stored program specifies the type of operation to be performed and the main memory storage locations of the data to be manipulated.
2. The operands are transferred to the operand storage registers a character at a time, beginning with the rightmost character in each operand.
3. Each pair of characters that enters the storage registers is combined in the adder. The result is stored in the main memory as specified by the program instruction. If a carry is generated, it is stored in the adder and combined with the next higher-order pair of characters.

Standard Processing Mode

The VLF processor performs arithmetical and logical operations as directed by the instructions of an internally stored program. Control circuitry within the processor selects, interprets, and executes these instructions. Normally, the instructions are executed sequentially. Branch instructions are provided, however, which make it possible to skip over a group of instructions or otherwise change the sequence of the program.

Interrupt Processing Mode

Sequential instruction execution is changed temporarily whenever the VLF processor is interrupted. Any one of four sources can "demand" access to the VLF processor by generating an interrupt signal, which turns on a processor interrupt indicator. Once an interrupt indicator is detected as being on, a hardware response is made: information concerning the current

[1] The contents of these registers are not accessible to the programmer.

status of the processor (including the setting of the sequence register) is stored, and a branch is made to a stored routine which identifies and services the demand. Thus, programmed tests need not be made to detect the presence of an interrupt condition — the process of detecting and responding to an interrupt signal is an automatic hardware function. After the stored service routine has been executed, control is returned to the interrupted routine at the point where the interruption occurred, and the previous status is restored. Two kinds of interrupts can occur in the system: external interrupts and an internal interrupt. A detailed description of interrupt functions and programming for interrupt processing is presented in Appendix E.

EXTERNAL INTERRUPTS

The three sources of external interrupts are:

1. Peripheral Control — The peripheral control connected to any Series 200 peripheral device can generate an interrupt signal under program control (peripheral control interruption is described in Appendix E). For instance, a communication control which services one or a number of communication lines and devices may generate a real-time demand on VLF processor time to handle a customer inquiry from a remote terminal. The current operations of the processor are temporarily interrupted so that the inquiry may be serviced. A routine to read the inquiry and to answer the question from a stored customer file is automatically executed, and a response is sent back to the terminal.
2. Operator's Console — The operator can interrupt the VLF processor by pressing the INTERRUPT button on the console. The source of such "on-site" interruptions is made available to the program by the execution of a single instruction at the beginning of the interrupt service routine.
3. Program Instruction — One instruction in the VLF repertoire, the Monitor Call instruction, is used to generate an interrupt condition. For programming convenience, the activation (or "calling") of the monitor program can be accomplished by means of this instruction.

INTERNAL INTERRUPTS

With the storage protection capability of the VLF processor, an internal interrupt condition, caused by attempted memory address violations or attempts to address nonexistent memory locations, can also occur. Internal interruptions are of lower priority than external interrupts, so that a processor executing an external interrupt service routine does not respond to an internal interruption until the routine is completed. Processing of internal interrupts is described in Appendix E.

Item-Mark Trapping Mode

The item-mark trapping mode, which can be set via the CAM instruction, causes the processor to treat and execute any instruction containing an item-marked op code as if it were a Change Sequencing Mode (CSM) instruction; this results in a transfer of control to an instruction stored at a prespecified location. This processor mode is used extensively in Liberator systems and can also be used to control program branching.

ADDRESSING

The main memory storage locations that contain the instructions and data of a program are identified to the machine by their particular main memory addresses. Control register length enables the VLF processor to have direct access to the 524,288 main memory characters. Due to the binary system used in addressing the storage locations within the VLF processor, an address portion of a machine-language instruction can occupy two, three, or four characters of memory. The number of character positions used is controlled by two instructions: the ADMODE assembly control statement, and the Change Addressing Mode (CAM) instruction. Any storage address can be specified in any addressing mode by having the processor prefix the address expressed in the instruction with a binary value previously set in an address register.

Three additional features that facilitate addressing are indexed and indirect addressing, and variable-length address interpretation.

The VLF processor contains main memory index registers which provide an automatic means for address modification without altering the instruction in which the address is modified. Indirect addressing enables a reference to be made to stored information via one or more intermediary addresses. Variable-length address interpretation refers to the ability of the VLF processor to operate in three different address interpretation modes, allowing the programmer to code instructions using either 2-character, 3-character, or 4-character addresses. This facility provides the flexibility necessary to allow the direct addressing of this large memory, while at the same time saving processing time and memory space when working in localized areas of memory.

An instruction is usually stored in a field of from 1 to 12 characters, depending on the format of the instruction and the mode of address assembly (2-, 3-, or 4-character). Figure 6-1 illustrates how a typical Add instruction appears when stored in the main memory. (Enclosing a character in a circle indicates that a word mark is associated with it.)

An instruction is addressed by specifying the op code (leftmost) location of the instruction. For instance, the address of the Add instruction in Figure 6-1 is 524. The machine reads an instruction from left to right until it senses a word mark. The extraction of the Add instruction below is stopped by the word mark associated with the op code of the next instruction in sequence.

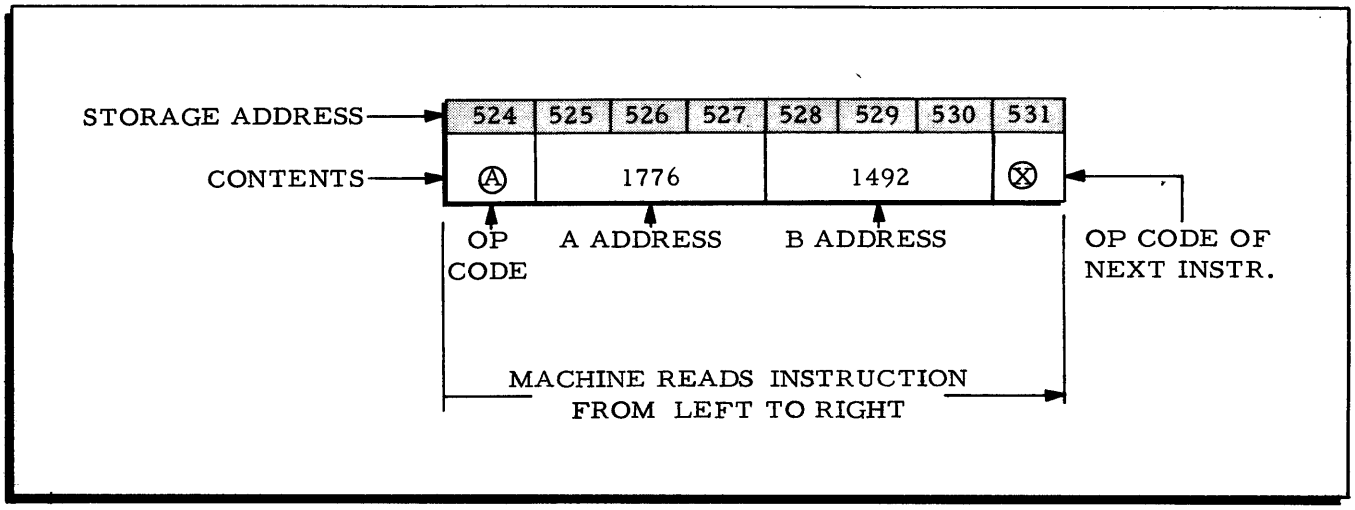


Figure 6-1. Typical Add Instruction.

A data field is normally defined in the following manner: the leftmost location in the field is indicated by a word mark; the rightmost location is specified in the A or B address of an instruction. The machine reads a data field from right to left until it senses the word mark associated with the leftmost character in the field.[1] For example, the A and B addresses in the instruction shown in Figure 6-1 could specify the data fields shown in Figure 6-2, below.

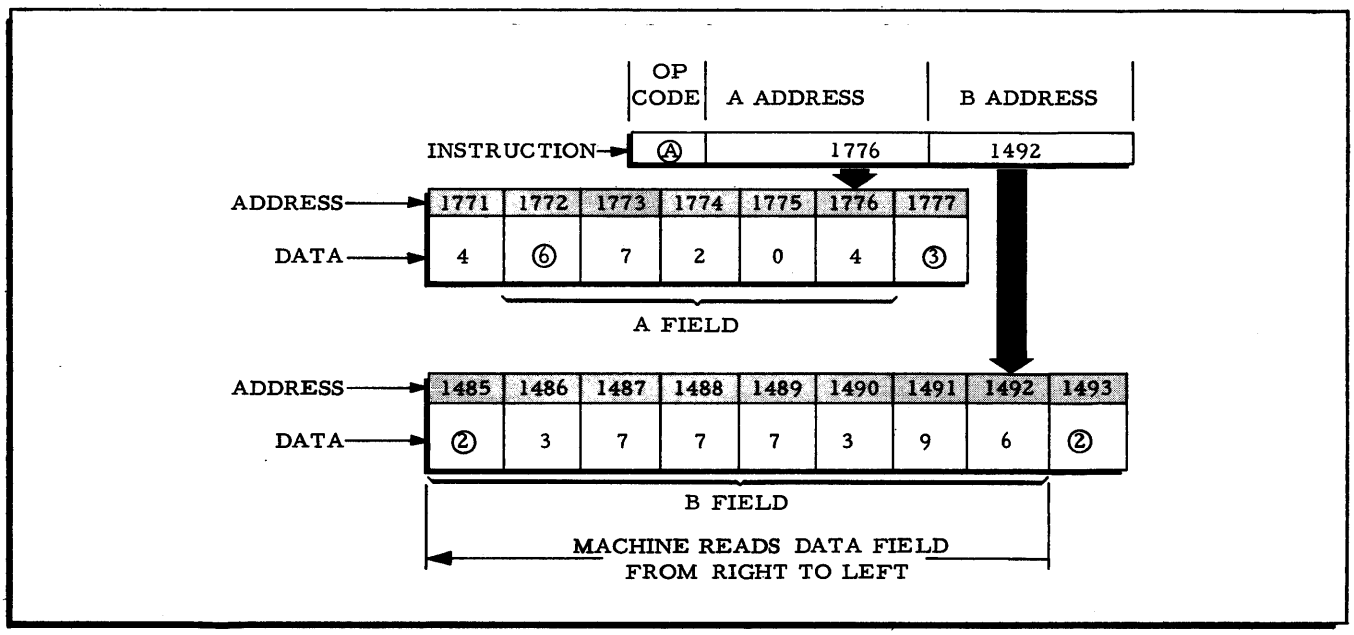


Figure 6-2. Extraction of Data Fields in Typical Add Instruction

[1] The Extended Move (EXM) instruction permits the reading of fields, items, and records in either direction.

An item is addressed by specifying either its leftmost or its rightmost character location in an address portion of an instruction (a variant character in the instruction specifies which character is being addressed). If the address of the leftmost character is specified, the machine reads the item from left to right; if the address of the rightmost character is specified, the machine reads the item from right to left. In either case, the operation terminates when an item mark is sensed.

A record is addressed by specifying its leftmost character location in an address portion of an instruction. The machine reads a record from left to right until it senses a record mark. Note that the contents of the character position containing a record mark are not considered as part of the record, except when the record is moved internally.

The direction in which the machine reads any of the above-mentioned groups is compatible with the manner in which the contents of the group are manipulated. For instance, a field used in an arithmetic operation is read from right to left because such operations combine fields character by character, starting with the low-order or "units" position in each field. Similarly, an instruction is read from left to right because the machine must interpret the op code before it can manipulate the operand(s).

Registers Used in Addressing

The processing of a stored-program instruction consists of two phases: the retrieval (or "extraction") of the instruction from main memory storage and the execution of the instruction. Six control memory registers are used to address the main memory during instruction processing. Four registers — SR, CSR, EIR, and IIR — are related to the selection of instructions in a program; the other two registers — AAR and BAR — control the transfer of information from one storage location to another by containing the address portions of an instruction.

SEQUENCE REGISTER (SR)

SR contains the address of the next sequential instruction character to be extracted from the memory during a program run. The contents of SR are incremented by one as each instruction character is extracted, so that SR contains the address of the next instruction's op code when one instruction has been completely extracted.

CHANGE SEQUENCE REGISTER (CSR)

The address of an op code can be stored in CSR.[1] A Change Sequencing Mode instruction will interchange the contents of SR and CSR and thereby cause the program to branch to the

[1] A Load Control Registers (LCR) instruction can be used to store the desired op code address.

instruction whose op code address was stored in CSR. At this point in the program, CSR will contain the address of the op code following the Change Sequence Mode instruction. In order to return to this op code (i.e., to the initial sequence of instructions), another Change Sequencing Mode instruction can be issued.

EXTERNAL INTERRUPT REGISTER (EIR)

EIR, like CSR, can be used to store the address of an op code.[1] This address and the contents of SR will be interchanged automatically when an external interrupt signal is received. (An external interrupt signal can be generated by a peripheral control, by the console, by master control, or by the Monitor Call instruction.) In order to return to the normal sequence of instructions that was interrupted, a Resume Normal Mode instruction (see page 6-97) can be issued.

INTERNAL INTERRUPT REGISTER (IIR)

The address of an op code can also be stored in IIR.[1] Certain operations are considered as "violations" of the VLF processor storage protect capability, (e.g., the attempt to write into an area of memory that is not assigned to the program making such an attempt). An internal interrupt signal is generated when such a violation attempt occurs, and the contents of IIR and SR are automatically interchanged. The Resume Normal Mode instruction is used to return to the interrupted program.

A-ADDRESS REGISTER (AAR)

AAR normally contains the A-address portion of an instruction (i.e., the storage address of the rightmost character of the A-operand data). This address is loaded into AAR during the extraction phase of processing. In the execution of instructions whose operands are fields or rightmost-addressed fields or items, the contents of AAR are decremented by one as each character in the A field is manipulated.[2] The contents of AAR are incremented by one as each character in a record or leftmost-addressed field or item is extracted.[3]

B-ADDRESS REGISTER (BAR)

Normally, the B-address portion of an instruction is loaded into BAR during the extraction phase. During the execution of most instructions, the contents of BAR are decremented by one

[1] A Load Control Registers (LCR) instruction can be used to store the desired op code address.

[2] A field can also be moved internally from left to right by the Extended Move (EXM) instruction. In this case, the address register is incremented.

[3] A record can also be moved internally from right to left by means of the Extended Move instruction. In this case, the address register is decremented.

as each character in the B field is extracted.[1] If the B operand is a record or a leftmost-addressed item, the contents of BAR are incremented by one as each character is extracted.[2]

The foregoing information can be summarized as follows:

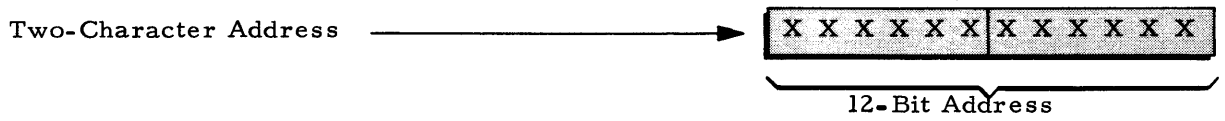
1. An instruction is read from left to right. As each character in the instruction is read, the contents of the sequence register are incremented by one.
2. A field is read from right to left.[1] As each character in a field is read, the contents of the corresponding address register are decremented by one.
3. A record is read from left to right.[2] As each character in a record is read, the contents of the corresponding current location counter are incremented by one.
4. An item can be read either from left to right or from right to left. As each character in an item is read, the contents of the corresponding address register are incremented by one if reading from left to right, or decremented by one if reading from right to left.

Addressing Modes

As stated before in this section, an instruction is normally stored in a field of from 1 to 12 characters, depending on the instruction's format and the programmed addressing mode. The op code is stored as a single 6-bit character. Variant characters or I/O control characters, if any, are each stored as single characters. The number of character locations in which each address portion is stored depends on the addressing mode selected by the programmer. This selection is made by means of a Change Address Mode instruction, with which the programmer specifies the 2-, 3-, or 4-character addressing mode.

TWO-CHARACTER ADDRESSING MODE

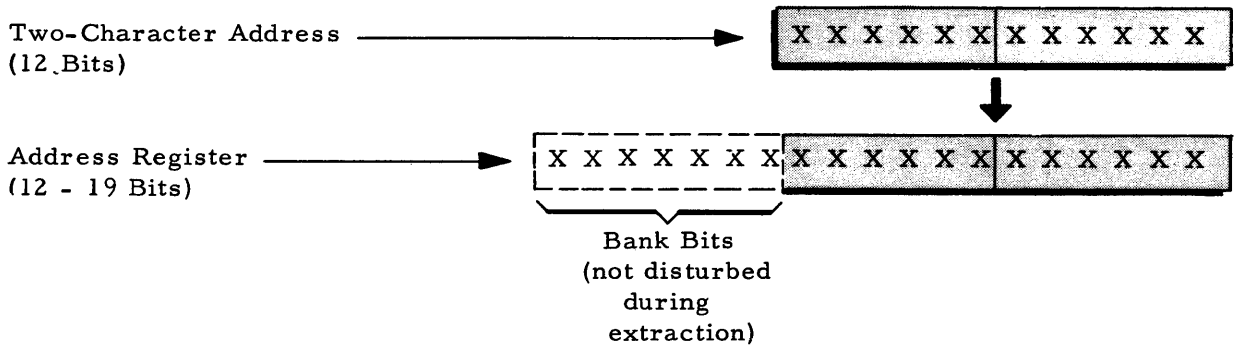
An operand address written in the 2-character addressing mode is stored in two consecutive character locations in memory. The stored address (a continuous 12-bit binary number) represents the address of a main memory location in the range 0 - $4,095_{10}$.



[1] A field can also be moved internally from left to right by the Extended Move (EXM) instruction. In this case, the address register is incremented.

[2] A record can also be moved internally from right to left by means of the Extended Move instruction. In this case, the address register is decremented.

During the extraction phase of instruction processing, the 2-character address is placed in the rightmost 12-bit positions of the address register (AAR or BAR). Any bits in the register to the left of the 2-character address are called "bank bits." Previous values in the bank bit positions of the register are not disturbed during instruction extraction.[1]



When the instruction is executed, the entire contents of the address register are interpreted as the operand address. Previous values in the bank bit positions, not disturbed during the extraction phase, are used to form the address of the operand during the execution phase. Thus, the bank bit values imply a base address to which the 12-bit address is added to form the actual operand address. If the bank bit values are all zeros, the 12-bit address is the actual operand address.

For example, a 2-character A address specifying location $4,000_{10}$ is extracted and placed in AAR. The second bank bit in AAR (bit position 14) contains a residual value of "1," representing a base address of $8,192_{10}$. When the instruction is executed, the entire contents of AAR ($8,192_{10} + 4,000_{10}$) specify the address of the A operand — location $12,192_{10}$.

As the contents of the address register are incremented or decremented during "internal" execution, bank bits are not disturbed.[2] If the 12-bit address in the rightmost positions of the

[1] The entire contents of an address register (bank bits + 12-bit address) are affected during the extraction of an instruction whose extraction path "duplicates A." Extraction of all other 2-character addresses affects only the rightmost 12 bits (see page 6-18).

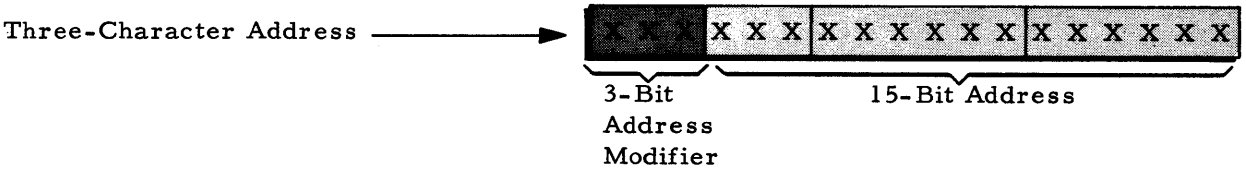
[2] "Internal execution" is defined as the incrementing or decrementing of address register contents during memory cycles allocated to the central processor. When peripheral transfer operations are performed, using memory cycles allocated to read/write channels, incrementing and decrementing of address register contents affect all bits of the register. Thus, addressing during peripheral transfer operations is continuous throughout the memory.

register becomes zero, a borrow from the first bank bit does not occur. Thus, the portion of memory which is addressable by a 1-character address is the 4,096-character "bank" specified by the base address.

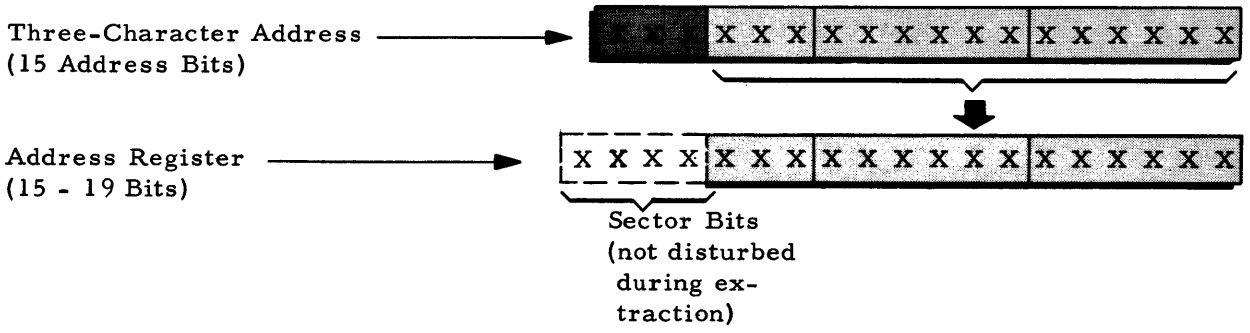
Neither indexed nor indirect addressing can be performed in the 2-character addressing mode.

THREE-CHARACTER ADDRESSING MODE

An operand address written in the 3-character addressing mode is stored in three consecutive character locations of the memory. The rightmost 15 bits of the stored address represent the address of a main memory location in the range 0 - $32,767_{10}$. The leftmost three bits, referred to as the "address modifier," specify whether the address is direct, indirect, or indexed (see "Address Modification," page 6-12).



During the extraction phase, the 15-bit address is placed in the rightmost bit positions of the operand address register. Any bits in the register to the left of these bit positions are called "sector bits." Previous values in the sector bit positions of the register are not disturbed during instruction extraction.[1]



[1] The entire contents of an address register (sector bits + 15-bit address) are affected during the extraction of an instruction whose extraction path "duplicates A" (described on page 6-10). Extraction of all other 3-character addresses affects only the rightmost 15 bits in the register.

When the instruction is executed, the entire contents of the address register are interpreted as the operand address. Previous values in the sector bit positions, not disturbed during the extraction phase, are used to form the address of the operand during the execution phase. Thus, the sector bit values imply a base address to which the 15-bit address is added to form the actual operand address. If the sector bit values are all zeros, the 15-bit address is the operand address.

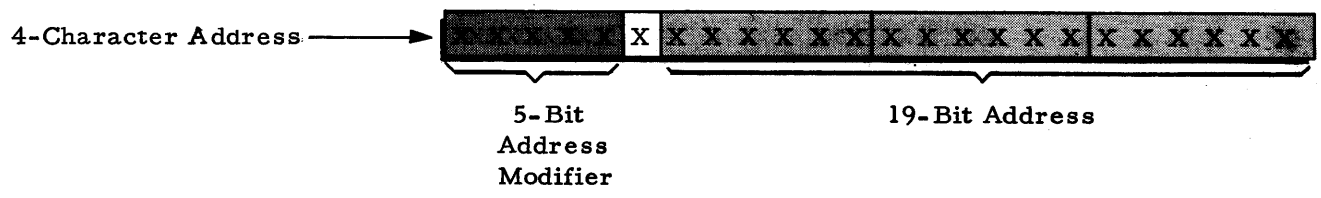
For example, a 3-character A address specifying location $12,000_{10}$ is extracted and placed in AAR. The first sector bit in AAR (bit position 16) contains the value "1," representing a base address of $32,768_{10}$. When the instruction is executed, the entire contents of AAR ($32,768_{10} + 12,000_{10}$) specify the address of the A operand — location $44,768_{10}$.

As the contents of the address registers are incremented or decremented during "internal" execution, sector bits are not disturbed. If the 15-bit address in the rightmost locations of the address register becomes zero, a borrow from the first sector bit does not occur. Thus, the largest portion of memory which is addressable by a 3-character address is the 32,768-character "sector" specified by the base address.

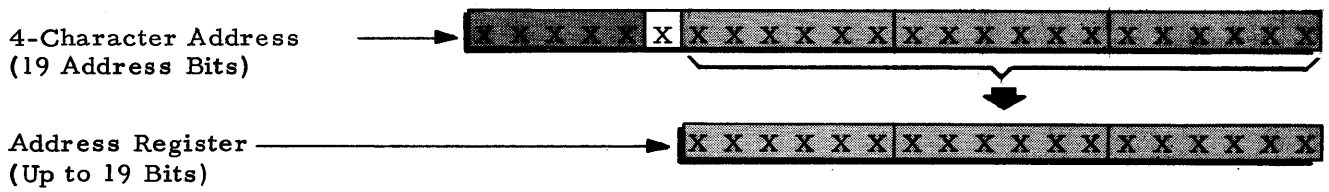
Addressing is continuous throughout the entire memory when a peripheral transfer operation is performed, as in the 2-character mode.

FOUR-CHARACTER ADDRESSING MODE

An operand address written in the 4-character addressing mode is stored in four consecutive character locations. The rightmost 19 bits represent a main memory address in the range 0 - $524,288_{10}$. The leftmost five bits - the "address modifier" — specify whether the address is direct, indirect, or indexed (see "Address Modification," page 6-12).



The 19-bit address is placed in the address register during the extraction phase. Thus, the entire contents of the address register are affected during the extraction of a 4-character address.



The entire contents of the register are interpreted as the operand address when the instruction is executed. As the contents of the operand address registers (AAR and BAR) are incremented or decremented during execution, all bits in the register are affected. Thus, addressing is continuous throughout the entire range of available memory (up to 524,288 locations) in the 4-character addressing mode.

Address Modification

Indirect and indexed addressing can be used to modify 3- or 4-character addresses in the VLF processor.

INDEX REGISTERS

Index registers are used to store values to be used for address modification during instruction execution. The portion of the VLF processor's index register complement usable by a program at any given time varies with the program's location in main memory and the addressing mode in use.

THREE-CHARACTER ADDRESS

The address modifier of a 3-character address (i.e., the leftmost three bits of the stored address) specifies whether the address is direct (000), indirect (111), or indexed (001 through 110).

Indirect Addressing

In previous examples in this section, an address portion of an instruction always specified the address of a data field in the main memory. This manner of addressing an operand is commonly referred to as direct, or "first level," addressing. In some instances, instead of specifying the location of a data field directly, it is more useful to be able to specify the storage location of another address, which in turn specifies the location of the desired data field. This manner of locating an operand is referred to as indirect, or "second-level," addressing.

A 3-character indirect address is specified by an address modifier of all one bits and refers to the leftmost storage location of another main memory address. The referent address can itself be direct, indirect, or indexed as specified by its address modifier. Thus, an

indirect address can specify another indirect address, and so on through any number of levels, or it can specify an indexed address.

Figure 6-3 shows the extraction of an Add instruction in which indirect addressing is specified in the A address and direct addressing is specified in the B address. Note that the A address (indirect) refers to the leftmost location of another main memory address. This address, in turn, specifies the location of the rightmost character in the A field. Note further that if the address modifier of location 1027 were not "000," the remainder of the stored address would be interpreted as an indexed or indirect address.

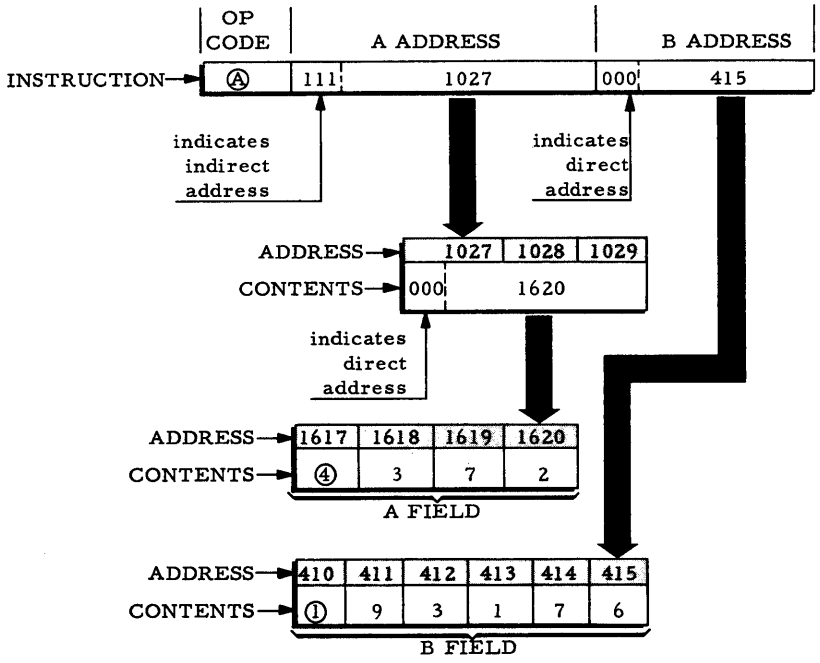


Figure 6-3. Extraction of 3-Character Indirect Address

Indexed Addressing

When indexed addressing is performed in the 3-character mode, the rightmost 15-bit contents of an index register are automatically added to the 15-bit contents of an index register are automatically added to the 15-bit address field in an instruction. Three variables must be defined in any indexing operation: (1) the index register to be used, (2) the address to be modified, and (3) the factor (referred to as an augment) to be added to the address. The index register to be used is specified in the address modifier of an address field. The address to

be modified can be stored in the same address field or it can be stored in the designated index register. If the address to be modified is stored in the address field, the augment is stored in the designated index register and vice versa.

Table 6-2 Index Register Addresses in 3-Character Addressing Mode

Index Register	Address Modifier	Storage Field	Address
X1	001	2 - 4 (+n)	4 (+n)
X2	010	6 - 8 (+n)	8 (+n)
X3	011	10 - 12 (+n)	12 (+n)
X4	100	14 - 16 (+n)	16 (+n)
X5	101	18 - 20 (+n)	20 (+n)
X6	110	22 - 24 (+n)	24 (+n)

n = contents of internal base relocation register.

The modification of an address occurs in its respective address register. For instance, if the B-address portion of an instruction is indexed, the modification is performed in BAR. This means that neither the original instruction stored in the main memory nor the contents of the index register is altered in any way.

Normal programming, such as a load or move operation, can be used to store a value in an index register. Similarly, the contents of an index register can be changed by using an instruction such as Binary Add or Binary Subtract. Note that since the index registers are located in the main memory, they can be used as normal storage locations when they are not being used for indexing operations.

Figure 6-4 illustrates how the Add instruction on page 6-13 would be extracted if indexed addressing were specified in the A-address portion of the instruction.

FOUR-CHARACTER ADDRESSING MODE

The address modifier in a 4-character address consists of the leftmost five bits of the address. The configuration of these bits specifies whether the address is direct (00000), indirect (10000), or indexed (00001 through 01111).

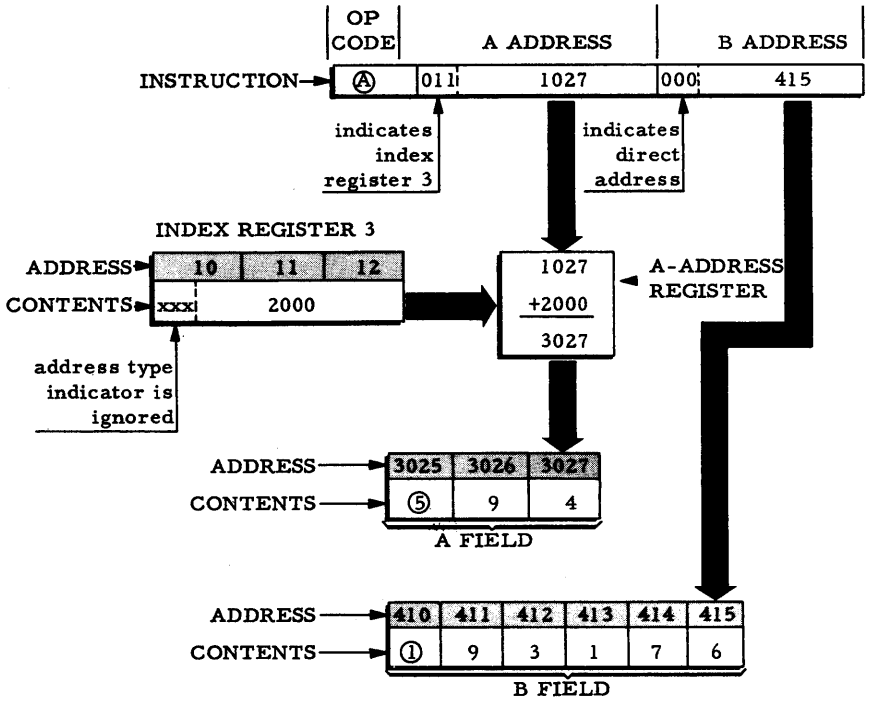


Figure 6-4. Extraction of Indexed Address in 3-Character Mode

Indirect Addressing

Indirect addressing in the 4-character addressing mode is performed similarly to that in the 3-character mode, except that:

1. a five-bit address modifier whose configuration is 10000 specifies indirect addressing; and
2. a 4-character address is extracted.

The method of coding a 4-character indirect address in assembly language is identical to that used for a 3-character indirect address.

Indexed Addressing

Four-character indexed addresses to be modified by index registers X1 through X15 are specified by an address modifier whose configuration is 00001 through 01111, respectively. The VLF processor provides 15 index registers for each program resident in core. The location of these index registers in memory is specified by the address modifier bits and the contents of the internal base relocation register. Register locations are shown in Table 6-3.

Table 6-3. Index Register Addresses in 4-Character Addressing Mode

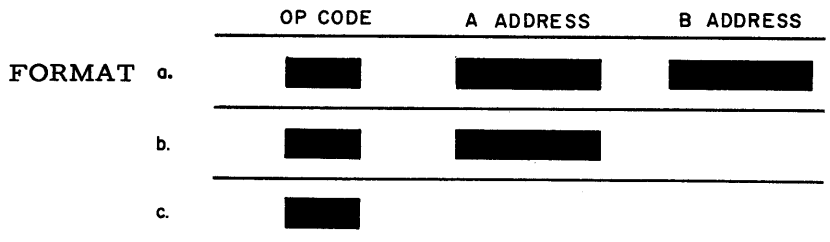
Index Register	Address Modifier	Storage Field	Address
X1	00001	1-4	4(+n)
X2	00010	5-8	8(+n)
X3	00011	9-12	12(+n)
X4	00100	13-16	16(+n)
X5	00101	17-20	20(+n)
X6	00110	21-24	24(+n)
X7	00111	25-28	28(+n)
X8	01000	29-32	32(+n)
X9	01001	32-36	36(+n)
X10	01010	37-40	40(+n)
X11	01011	41-44	44(+n)
X12	01100	45-48	48(+n)
X13	01101	49-52	52(+n)
X14	01110	53-56	56(+n)
X15	01111	57-60	60(+n)

When indexed addressing is performed in the 4-character mode, the 19 bits of the specified index register are added to the address field of the instruction.

The extraction of a Subtract instruction written in the 4-character addressing mode is shown in Figure 6-5. Indirect addressing is specified in The A address, and indexed addressing is specified in the B address.

Explicit Addressing, Implicit Addressing, and Chaining

Consider the three instruction formats below.



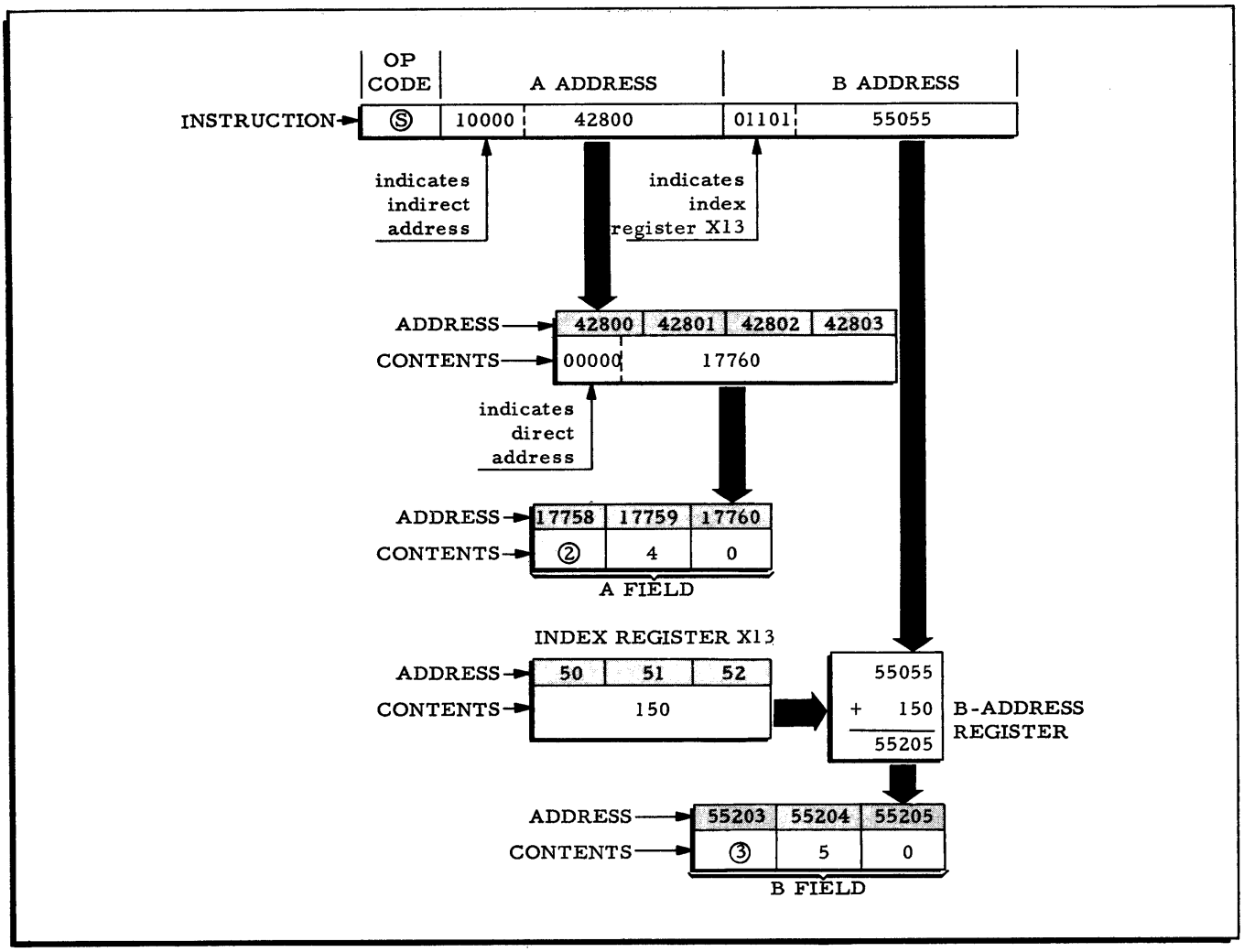


Figure 6-5. Extraction of Indirect and Indexed 4-Character Addresses

Format a. corresponds to the instructions used in the preceding illustrations. The significant feature of this format is that the addresses of both the A and B data fields are explicitly specified in the instruction. For this reason, the data fields are said to be "explicitly addressed." In general, whenever the programmer writes the address of a data field on his coding sheet, he is explicitly addressing that data field (see Figure 6-6).

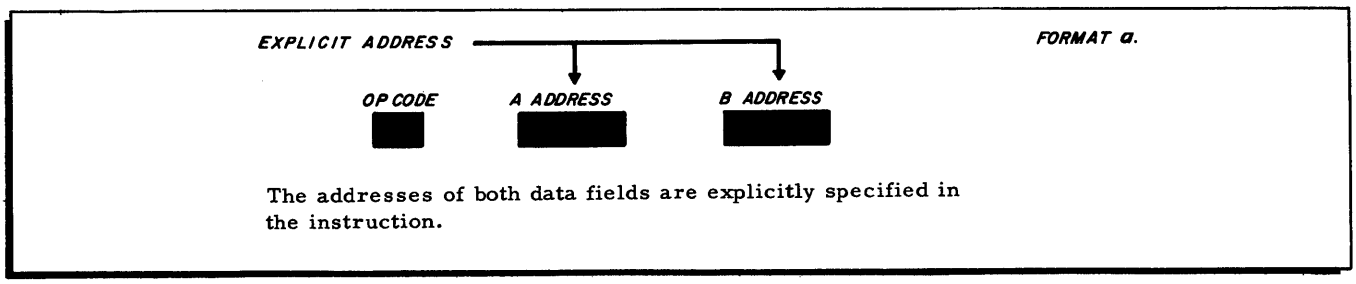


Figure 6-6. VLF Processor Instruction Format a

Format b. has two possible interpretations (see Figure 6-7):

1. Ten VLF processor instructions coded in format b. cause the A address to be loaded into both AAR and BAR.[1] Thus, although the B- address portion of the instruction is omitted, the B field is explicitly addressed by the A-address portion. The extraction path of these instructions is said to "duplicate A," since the contents of AAR are duplicated in BAR.

2. The A address of 19 instructions is loaded into AAR only, leaving BAR undisturbed. An omitted B address in any of these instructions implies that the previous contents of BAR will be used as the address of the B field. For this reason, the B field is said to be "implicitly addressed," and the extraction path of these instructions "preserves B."

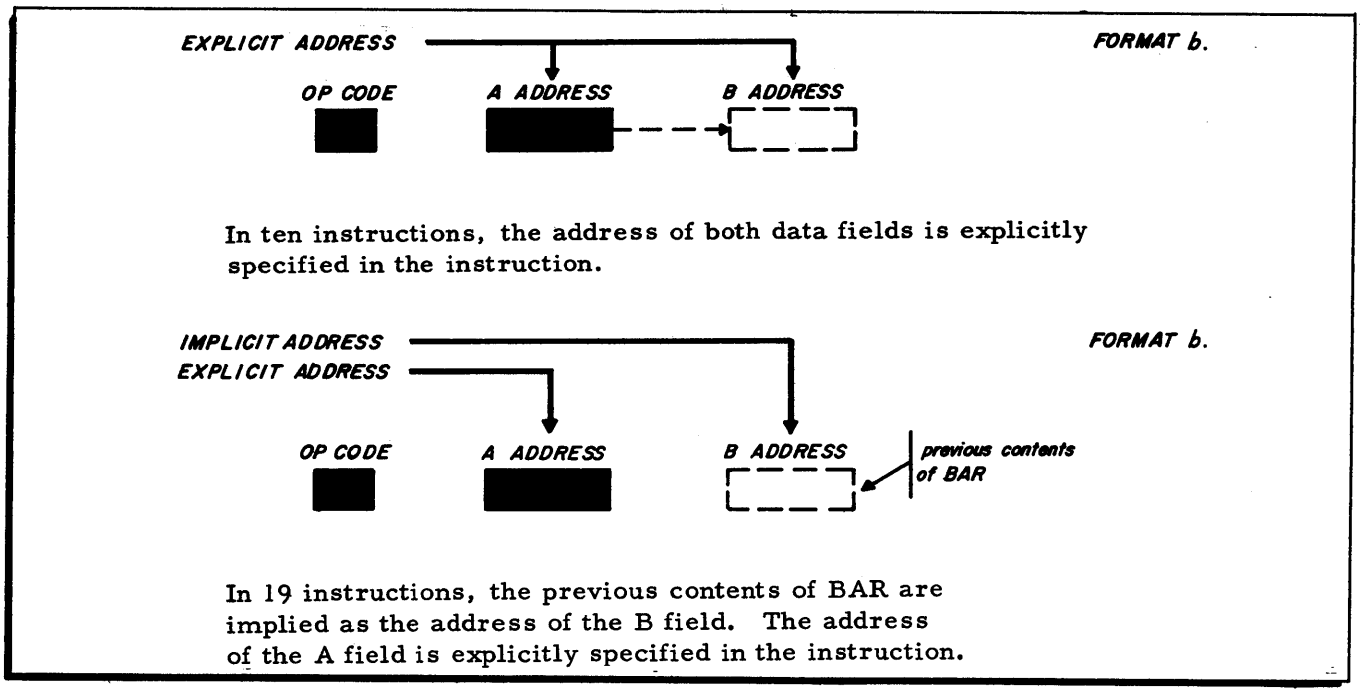
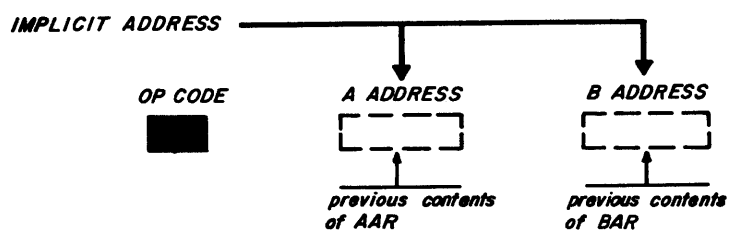


Figure 6-7. VLF Processor Instruction Format b

In format c., both data fields are implicitly addressed. The previous contents of AAR are used as the address of the A field, and the previous contents of BAR are used as the address of the B field (see Figure 6-8).

Implicit addressing is extremely useful in situations where it is desired to perform a series of operations on data fields that are in consecutive storage locations. The use of implicit address reduces both the time required to perform the operations and the number of memory locations required to store the instructions.

[1] The entire contents of AAR are loaded into BAR during extraction, so that all bit positions in BAR are identical to those in AAR.



The addresses of both data fields are implied in the instruction.

Figure 6-8. VLF Processor Instruction Format c

Instructions which require a variant character can also be chained, in most cases, by using the previous contents of the variant register. (The variant register is a single-character, internal register into which the variant character of an instruction is loaded during execution.)

In the VLF processor, variant characters can be chained by an instruction coded in any format (i.e., format a., b., or c. shown on page 6-16). The previous contents of the variant register are not disturbed by the processing of an instruction which does not contain a variant character.

Chaining is not limited to sequential operations having the same op code. The only condition that must be met is that an instruction must be met is that an instruction must leave the contents of AAR, BAR, and, if required, the variant register such that they satisfy the addressing requirements of the next instruction in sequence.

To enable the programmer to chain instructions wherever possible, the instruction descriptions presented later in this section include a table showing the contents of the address registers after the instruction has been executed. Also, Appendix C denotes whether each instruction in the machine complement can or cannot be chained.

EXTENDED MULTIPROGRAMMING

Extended multiprogramming in the VLF processor provides five basic capabilities required in a multiprogramming environment. These are:

1. Base relocation,
2. Storage protection with base relocation,

3. Interrupt masking,
4. Instruction timeout, and
5. 8-bit transfer capability.

Base Relocation

The storage protection offered by extended multiprogramming is made possible using base relocation in conjunction with storage protection. Base relocation is in effect when the relocation indicator is set (via the Store Variant and Indicators (SVI) and Restore Variant and Indicators (RVI) instructions) and the processor is in the standard (noninterrupt) mode (see Appendix F).

Storage Protection with Base Relocation

Storage protection with base relocation places a barrier above and below the area of memory where the active program is to operate, to prevent it from altering the contents of the rest of memory. The lower barrier is specified by the contents of the base relocation register (BRR), which is loaded and stored via Load Index/Barricade Register (LIB) and Store Index/Barricade Register (SIB) instructions (see LIB and SIB instructions, pages 6-83 and 6-85). When relocation is in effect, the BRR is loaded with the bank address of the lowest memory bank (4,096 characters) available to standard mode programs. The BRR is added to each processor memory address transmitted to memory by a standard mode program. This prevents a standard mode program from addressing a memory bank below that specified by the BRR. The upper barrier is specified by the contents of the index/barricade register (IBR). When storage protection is in effect and an attempt is made to write into memory at an address greater than that stored in the IBR, a protection violation occurs resulting in an internal interrupt (see Appendix F). The IBR contains the integral multiple of the 4,096 characters in which a program is stored.

A monitor program keeps track of the locations of the various programs stored in memory and, via the settings of the BRR and the IBR, can relocate references to any number of 4096-character banks of memory. Thus, while there may be any number of programs stored in memory, only one program is active at any one time and all other programs are protected from the active program when storage protection is in effect. When, as the result of an interrupt, the monitor program activates a different program, it simply alters the settings of the BRR and the IBR to make available a different portion of memory.

Since all VLF processor memory references are relocated via the BRR when relocation is in effect, index registers X1 through X15 effectively reside in the 4096-character bank of memory specified by the BRR. The location of index registers Y1 through Y15 is also dependent on the setting of the relocation indicator. When relocation is activated, the Y index registers are also located in the 4096-character bank specified by the BRR, where they become identical

to index registers X1 through X15. When relocation is in effect, each program stored, including the monitor program, has its own set of 15 index registers when it is the active program.

External Interrupt Masking

Each input/output (I/O) sector has associated with it a one-bit mask. This mask is stored and set by Store Variant and Indicators (SVI) and Restore Variant and Indicators (RVI) instructions, respectively (see SVI and RVI instructions, pages 6-90, and 6-94). When the mask for a sector is zero, interrupts from sources in that sector are accepted and processed in the manner specified in Appendix E of this manual. When the mask for a sector is one, then interrupts are held until the mask is altered or the interrupt function is reset. Console interrupt, Monitor Call, and external interrupts set by master control are never masked.

Instruction Timeout

It is possible for an instruction in a program to enter an infinite extraction loop or extensive execution loop. To prevent this from occurring, a timeout function is provided which allows a maximum time limit to be placed on the extraction and the execution of any one instruction. The timeout function is reset to zero and begins timing every time the processor starts to extract or execute a new instruction. If the timeout allow function is on and the processor is in the standard mode when the time interval elapses, then the instruction being extracted or executed is terminated, the timeout function goes on, and an internal interrupt is generated. This function guarantees that the monitor program will at some specified time regain control of the system. The timeout function is enabled by a timeout allow function, which is set and reset by SVI and RVI instructions.

8-Bit Transfer Capability

This capability allows the VLF processor to transfer data between peripheral controls and memory in either 6- or 8-bit format, as specified in the Peripheral Data Transfer (PDT) instruction.

1. The 6-bit mode is the standard data transfer mode used in Series 200 Central Processors. In this mode, only data is transferred between memory and peripheral controls. Punctuation is preserved in memory.
2. The 8-bit mode is used in those applications where an 8-bit transfer is desired between the central processor and a peripheral control. In this mode of operation, data and punctuation are transferred between the central processor and peripheral controls. Record marks in memory do not terminate data transfer in this mode.

When in the 8-bit mode, the number of 8-bit character transfers to be performed is determined by a count field in the PDT instruction or by control characters associated with the peripheral controls.

INSTRUCTIONS

The VLF processor operates under the direction of instructions in the stored program. For descriptive purposes, these instructions are classified into six functional categories: (1) Arithmetic; (2) Logic; (3) Control; (4) Interrupt Control; (5) Editing; and (6) Input/Output.

All instructions are described in the following standard format:

Title:	The title describes the instruction. It appears in the left-hand margin of a page, along with the mnemonic operation code used in the symbolic programming language.
Format:	This is a tabular representation of all formats which may be used when coding the instruction.
Function:	The function of the instruction is described in terms of each format in which it can be coded.
Word Marks:	The effect of word marks with regard to data fields is specified.
Address Registers after Operation:	The contents of the address registers are indicated for each of the instruction's formats.
Notes:	This is additional information pertaining to the operation.
Examples:	Practical applications of the instruction in its various formats are described and illustrated as symbolic program entries.

Table 6-4, below, lists the abbreviations and symbols used in the description of the instructions. Those symbols used only with specific instructions are preceded by the title of the instruction to which they pertain.

Table 6-4. Symbology Used in Model 8200 Instruction Descriptions

SYMBOL	MEANING
A	A address of the instruction
B	B address of the instruction
(A)	Contents of the A-address field
(B)	Contents of the B-address field
Ni	Number of characters in the instruction
Na	Number of characters in the A field
Nb	Number of characters in the B field
Nw	Number of characters in the A or B field, whichever is smaller
NXT	Address of next sequential instruction
JI	Address of next instruction if a branch occurs
Ap	The previous setting of the A-address register
Bp	The previous setting of the B-address register

Table 6-4 (cont). Symbology Used in Model 8200 Instruction Descriptions

SYMBOL	MEANING
AAR	A-address register (AAR) = the contents of the A-address register
BAR	B-address register (BAR) = the contents of the B-address register
	Divide
Ndd	Number of digits in the dividend
	Move and Translate
Nct	Number of characters translated
	Move Item and Translate
Nut	Number of information units translated
CSRp	Previous contents of the change sequence register (CSR)
NAu	Number of 6-bit character locations occupied by each A-item information unit (1 or 2)
NBu	Number of 6-bit character locations occupied by each B-item information unit (1 or 2)
	Table Lookup
Lta	The location in the table immediately to the left of the argument (or short field) that terminated the search
V	Variant character
SR	Sequence Register

In all instruction descriptions that follow, the op code is expressed in octal.

ARITHMETIC INSTRUCTIONS

Both decimal and binary arithmetic operations can be performed by the VLF processor, using the following eight instructions.

DECIMAL ADD
DECIMAL SUBTRACT
BINARY ADD
BINARY SUBTRACT
ZERO AND ADD
ZERO AND SUBTRACT
DECIMAL MULTIPLY
DECIMAL DIVIDE

Decimal arithmetic instructions treat their operands as signed numeric data. Normal algebraic sign control is in effect during the execution of these instructions. Any zone bit configuration other than 10 in the rightmost character of a decimal field causes the field value to be interpreted as positive; 10 indicates a negative field value.

If the content of A is algebraically larger than the content of B, a recomplement cycle is performed automatically to convert the result to its true form.

An indicator is set at the completion of each decimal arithmetic operation to indicate the presence or absence of a zero result. A different indicator is set if overflow is sensed. The status of either of these indicators can be tested by a subsequent programmed instruction.

In binary arithmetic operations, the operands are treated as unsigned binary numbers; overflow and zero balance are disregarded.

A | DECIMAL ADD

FORMAT

	Op Code	Address Field	Address Field
a.	36	A	B
b.	36	A	
c.	36		

FUNCTION

Format a: The signed decimal data in the A field is added algebraically to the signed decimal data in the B field. The result is stored in the B field.

Format b: The signed decimal data in the A field is added to itself. The result is stored in the A field.

Format c: The signed decimal data specified by (AAR) is added algebraically to the signed decimal data specified by (BAR). The result is stored in the B field.

WORD MARKS

Format a: The B operand must have a defining word mark that terminates the operation. The A operand must have a word mark only if it is shorter than the B operand. In this case, transmission of data from the A operand stops after the A-operand word mark is sensed. If the A field is longer than the B field, the high-order characters of the A field that exceed the field length defined by the B-operand word mark are not processed.

Format b: The A operand must have a defining word mark.

Format c: The B operand must have a defining word mark. The A operand must have a word mark only if it is shorter than the B operand.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-Nw	B-Nb
Format b:	NXT	A-Na	A-Na
Format c:	NXT	Ap-Nw	Bp-Nb

NOTES

1. The algebraic sign control for the add operation is shown below.

A-FIELD SIGN	+	-	+	-
B-FIELD SIGN	+	-	-	+
TYPE OF ADD	True	True	Comp	Comp
SIGN OF RESULT	Sign of B field		Normalized sign of A or B field, whichever is greater (- = 10, + = 01)	

2. All zone bits in the result field are set to zeros except for the units position (i.e., the sign of the result).

3. The Decimal Add instruction treats both operands as signed decimal data. The instruction will produce ambiguous results if used to manipulate non-decimal data. Particularly, if the four numeric bits of any character have a binary numeric value of 12 or more (octal 14, 15, 16, and 17), the character is treated as if it were a zero. The two remaining cases (octal 12 and 13) are unspecified.

4. The overflow and zero balance indicators are set by an add operation.

EXAMPLE

Add Bond Deductions to Total Deductions.

Description	Tag
Bond Deductions	BDED
Total Deductions	TDED

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ . ___ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1				A	BDED, TDED	
2						
3						
4						
5						
6						

S	DECIMAL SUBTRACT

FORMAT

	Op Code	Address Field	Address Field
a.	37	A	B
b.	37	A	
c.	37		

FUNCTION

Format a: The signed decimal data in the A field is subtracted algebraically from the signed decimal data in the B field. The result is stored in the B field.

Format b: The signed decimal data in the A field is subtracted from itself. The result is stored in the A field. If the A-field sign is minus, the result is a minus zero. If the A-field sign is plus, the result is a plus zero (with normalized sign).

Format c: The signed decimal data specified by (AAR) is subtracted algebraically from the signed decimal data specified by (BAR). The result is stored in the B field.

WORD MARKS

Format a: The B operand must have a defining word mark. The A operand must have a word mark only if it is shorter than the B operand. In this case, transmission of data from the A operand stops after the A-operand word mark is sensed. If the A field is longer than the B field, the high order characters of the A field that exceed the field length defined by the B operand word mark are not processed.

Format b: The A operand must have a defining word mark.

Format c: The B operand must have a defining word mark. The A operand must have a word mark only if it is shorter than the B operand.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-Nw	B-Nb
Format b:	NXT	A-Na	A-Na
Format c:	NXT	Ap-Nw	Bp-Nb

NOTES

1. Algebraic sign control for the subtract operation is shown below.

A-FIELD SIGN	+	-	+	-
B-FIELD SIGN	+	-	-	+
TYPE OF ADD	Comp	Comp	True	True
SIGN OF RESULT	Normalized sign of A or B field whichever is greater (- = 10, + = 01)		Sign of B field	

2. All zone bits in the result field are set to zeros except for the units position (i.e., the sign of the result).

3. This instruction treats both operands as signed decimal data. It will produce ambiguous results if used to manipulate non-decimal data. Particularly, if the four numeric bits of any character have a binary numeric value of 12 or more (octal 14, 15, 16, and 17), the character is treated as if it were a zero. The two remaining cases (octal 12 and 13) are unspecified.

4. The overflow and zero balance indicators are set by a decimal subtract operation.

EXAMPLE

Subtract the contents of the 5-character fields starting at location 940, 945, 950, and 955 from the contents of the 8-character fields starting at locations 648, 656, 664, and 672.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ____ OF ____

	CARD NUMBER	TYPE	MAR	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1					S	955,672	
2					S		
3					S		
4					S		
5							
6							

BA	BINARY ADD

FORMAT

	Op Code	Address Field	Address Field
a.	34	A	B
b.	34	A	
c.	34		

FUNCTION

Format a: The data in the A field is added in binary fashion, character by character, to the data in the B field. The result is stored in the B field.

Format b: The data in the A field is added, character by character, to itself. The result is stored in the A field.

Format c: The data specified by (AAR) is added, character by character, to the data specified by (BAR). The result is stored in the B field.

WORD MARKS

Format a: The B operand must have a defining word mark that terminates the operation. The A operand must have a word mark only if it is shorter than the B operand. In this case, the transmission of data from the A field stops after the A-operand word mark is sensed. If the A field is longer than the B field, the high-order characters of the A field that exceed the field length defined by the B-operand word mark are not processed.

Format b: The A operand must have a defining word mark.

Format c: The B operand must have a defining word mark. The A operand must have a word mark only if it is shorter than the B operand.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-Nw	B-Nb
Format b:	NXT	A-Na	A-Na
Format c:	NXT	Ap-Nw	Bp-Nb

NOTES

1. The overflow and zero balance indicators are not set by a binary add operation.
2. Format b. of the BA instruction has the effect of doubling the value stored in the A field; i. e., it shifts the contents of the A field one bit position to the left.

EXAMPLE

Modify the B address of the instruction tagged B7 by the value stored in the location tagged TEN (assuming the use of the 2-character addressing mode).

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ____ PAGE ____ OF ____

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62 63 ... 80
1				BA	TEN, B7+4
2					
3					
4					
5					

BS	BINARY SUBTRACT

FORMAT

	Op Code	Address Field	Address Field
a.	35	A	B
b.	35	A	
c.	35		

FUNCTION

Format a: Each 6-bit character in the A field is converted to its ones complement and added, in binary fashion, character by character, to the data in the B field. A simulated carry is added with the characters in the units position. The result is stored in the B field.

Format b: Each 6-bit character in the A field is converted to its ones complement and added, character by character, to itself. A simulated carry is added with the characters in the units position. In effect, this format of the Binary Subtract instruction replaces the contents of the A field with zeros.

Format c: Each 6-bit character specified by (AAR) is converted to its ones complement and added, character by character, to the data specified by (BAR). A simulated carry is added with the characters in the units position. The result is stored in the B field.

WORD MARKS

Format a: The word mark in the B operand terminates the operation. The A operand must have a word mark only if it is shorter than the B operand. In this case, transmission of data from the A field stops after the A-operand word mark is sensed. If the A operand is longer than the B operand, the characters of the A operand that exceed the field length defined by the B-operand word mark are not processed.

Format b: The A operand must have a defining word mark.

Format c: The B operand must have a defining word mark. The A operand must have a word mark only if it is shorter than the B operand.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-Nw	B-Nb
Format b:	NXT	A-Na	A-Na
Format c:	NXT	Ap-Nw	Bp-Nb

NOTES

1. The overflow and zero balance indicators are not set by a binary subtract operation.

2. Formats a. and c. can produce negative results. A negative result is stored in the B field in the twos-complement form. In this case, the absolute numerical value of the result can be obtained by recomplementing the result stored in the B field. A negative result is detected only if the programmer provides appropriate coding to ascertain whether or not operands will produce such a result.

EXAMPLE

Zero the field starting at location TOTAL.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

	CARD NUMBER	TYPE	MAR	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1					BS	TOTAL	
2							
3							
4							

NOTE: Zone bits as well as numeric bits are cleared to zero by this operation.

ZA | ZERO AND ADD

FORMAT

	Op Code	Address Field	Address Field
a.	16	A	B
b.	16	A	
c.	16		

FUNCTION

Format a: The data in the A field is transferred, character by character, right to left, to the B field. Zone bits in the B field are set to zero in all positions except the units position. The sign of the result field is based on the sign of the A field (see note 1). If the high-order character of the A field is transferred before the operation terminates, the remaining B-field characters are cleared to zeros.

Format b: The data in the A field is converted to an all-numeric format; i.e., the zone bits of all positions in the field except the units position are set to zero. The result remains in the A field. The sign of the A field is not changed by the operation (see note 1).

Format c: The data specified by (AAR) is transferred to the field specified by (BAR). Zone bits in the B field are set to zero in all positions except the units position. The sign of the result field is based on the sign of the A field (see note 1). If the high-order character of the A field is transferred before the operation terminates, the remaining B-field characters are cleared to zeros.

WORD MARKS

Format a: The B operand must have a defining word mark. The A operand must have a word mark only if it is shorter than the B operand. In this case, transfer of data from the A operand stops after the A-operand word mark is sensed. If the A field is longer than the B field, the high-order characters of the A field that exceed the field length defined by the B-operand word mark are not processed.

Format b: The A operand must have a defining word mark.

Format c: The B operand must have a defining word mark. The A operand must have a word mark only if it is shorter than the B operand.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-Nw	B-Nb
Format b:	NXT	A-Na	A-Na
Format c:	NXT	Ap-Nw	Bp-Nb

NOTES

1. A plus sign in the units position of the result field is always expressed in its normalized form (01).
2. B-field punctuation is not changed by this operation.
3. This instruction does not set the overflow and zero balance indicators.

EXAMPLE

Transfer the contents of the field tagged ORATE to the field tagged NRATE, setting all zone bits in NRATE (except in the units position) to zeros.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1				ZA	ORATE, NRATE	
2						

ZS | ZERO AND SUBTRACT

FORMAT

	Op Code	Address Field	Address Field
a.	17	A	B
b.	17	A	
c.	17		

FUNCTION

Format a: The data in the A field is transferred to the B field with the opposite sign. Zone bits in the B field are set to zeros in all positions except the units position. If the high-order character of the A field is transferred before the operation terminates, the remaining B-field characters are cleared to zeros.

Format b: The data in the A field is converted to an all-numeric format; i.e, the zone bits of all positions in the field except the units position are set to zero. The result remains in the A field with its sign reversed.

Format c: The data specified by (AAR) is transferred with the opposite sign to the field specified by (BAR). Zone bits in the B field are set to zero in all positions except the units position. If the high-order character of the A field is transferred before the operation terminates, the remaining B-field characters are cleared to zeros.

WORD MARKS

Format a: The B operand must have a defining word mark. The A operand must have a word mark only if it is shorter than the B operand. In this case, transfer of data from the A operand stops after the A-operand word mark is sensed. If the A field is longer than the B field, the high-order characters of the A field that exceed the field length defined by the B-operand word mark are not processed.

Format b: The A operand must have a defining word mark.

Format c: The B operand must have a defining word mark. The A operand must have a word mark only if it is shorter than the B operand.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-Nw	B-Nb
Format b:	NXT	A-Na	A-Na
Format c:	NXT	Ap-Nw	Bp-Nb

NOTES

1. A plus sign in the units position of the result field is always expressed in its normalized form (01).
2. B-field punctuation is not changed by this operation.
3. This instruction does not set the overflow and zero balance indicators.

EXAMPLE

Change the sign of the data in the field tagged PROFIT.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ____ OF ____

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
				ZS	PROFIT	

M | DECIMAL MULTIPLY

FORMAT

	Op Code	Address Field	Address Field
a.	26	A	B
b.	26	A	
c.	26		

FUNCTION

Format a: The signed decimal integer in the A field is multiplied by the signed decimal integer in the leftmost locations of the B field. The product is stored, right-justified, in the B field.

Format b: The signed decimal integer in the A field is multiplied by the signed decimal integer in the leftmost locations of the field specified by the contents of the B-address register. The product is stored, right-justified, in the B field.

Format c: The signed decimal integer in the field specified by (AAR) is multiplied by the signed decimal integer in the leftmost locations of the field specified by (BAR). The product is stored, right-justified, in the B field.

WORD MARKS

Formats a, b, and c:

Word marks are required in the high-order locations of both the A and B fields. All other B-field locations must not contain word marks.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-Na	B-Nb
Format b:	NXT	A-Na	Bp-Nb
Format c:	NXT	Ap-Na	Bp-Nb

NOTES

1. The A address of a Decimal Multiply instruction specifies the units position of the multiplicand. The B address specifies a location which is Na+1 locations to the right of the multiplier, since the B field must contain the multiplier plus enough additional locations (to the right of the multiplier) to provide for the development of the product. Thus, the total number of character locations in the B field must be one greater than the sum of the number of characters in the multiplicand and the multiplier. For example, in a multiplication operation involving a 3-character multiplier and a 5-character multiplicand, 9 positions (5+3+1) must be provided in the B field.

2. Algebraic sign control for the multiply operation is shown below. The sign of the product is expressed in its normalized form (- = 10, + = 01).

Sign of Multiplicand	+	-	+	-
Sign of Multiplier	+	-	-	+
Sign of Product	+	+	-	-

3. The product is stored, right-justified, in the entire B field, with the unused high-order positions of the B field cleared to zeros. As a result of the operation, the multiplier (initially stored in the B field) is destroyed. Therefore, if the multiplier is to be used more than once, it should be preserved in another storage field.

4. The zero balance indicator is turned ON if the product of the multiply operation is equal to zero; otherwise, the indicator is turned OFF by the operation.

5. This instruction treats both operands as signed decimal data. It will produce ambiguous results if used to manipulate non-decimal data. Particularly, if the four numeric bits of a character have a binary numeric value of 12 or more (octal 14, 15, 16, or 17), the character is treated as if it were a zero. The two remaining cases (octal 12 and 13) are unspecified.

EXAMPLE

Multiply the 5-character field tagged CAND by the 3-character field whose rightmost character location is six (5+1) less than the location tagged PROD. Store the result, right-justified, in PROD.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

CARD NUMBER	TYPER	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1				M	CAND, PROD	
2						
3						

D | DECIMAL DIVIDE

FORMAT

	Op Code	Address Field	Address Field
a.	27	A	B
b.	27	A	
c.	27		

FUNCTION

Format a: The signed decimal integer whose leftmost location is B is divided by the signed decimal integer in the A field. The quotient is stored in the leftmost locations of the B field; the remainder is stored in the rightmost locations of the B field. [1]

Format b: The signed decimal integer whose leftmost location is specified by (BAR) is divided by the signed decimal integer in the A field. The quotient is stored in the leftmost locations of the B field; the remainder is stored in the rightmost locations of the B field.

Format c: The signed decimal integer whose leftmost location is specified by (BAR) is divided by the signed decimal integer in the field specified by (AAR). The quotient is stored in the leftmost locations of the B field; the remainder is stored in the rightmost locations of the B field.

WORD MARKS

Formats a, b, and c:

The A operand (the divisor) must contain a word mark. The B field may contain a word mark.

ADDRESS REGISTERS AFTER OPERATION (WHEN DIVISOR IS NOT EQUAL TO ZERO)

	SR	AAR	BAR
Format a:	NXT	A-Na	B-Na + Ndd -3
Format b:	NXT	A-Na	Bp-Na + Ndd -3
Format c:	NXT	Ap-Na	Bp-Na + Ndd-3

When the divisor is equal to zero, the contents of the address registers are unspecified (see note 1).

NOTES

1. If the divisor is equal to plus or minus zero, the overflow indicator is turned ON, division is not performed, and no memory locations are changed.
2. The length of the B field is determined by adding 1 to the sum of the number of character locations in the divisor and the dividend (B-field length = 1 + length of divisor + length of dividend).
3. The A field (divisor) can be signed or unsigned; if it is unsigned, the divisor is assumed to be positive.
4. The dividend must contain a normalized sign (- = 10, + = 01) in the units position. The zone bits of all other characters in the dividend must be zeros. The proper signing of the dividend is therefore insured if the dividend is moved into the B field by a Zero and Add instruction.

[1] Note that the B field in a divide operation does not define the B operand, but is a group of storage locations within which the B operand (the dividend) is contained.

5. All high-order locations of the B field which are not occupied by the dividend must contain zeros when division begins. These zeros can be automatically inserted if the Zero and Add instruction is used to move the dividend into the B field as mentioned above.

6. The sign of the quotient follows algebraic sign rules as shown below. The sign of the remainder is the original sign of the dividend.

Sign of divisor	+	+	-	-
Sign of dividend	+	-	+	-
Sign of remainder	+	-	+	-
Sign of quotient	+	-	-	+

7. This instruction treats both operands as signed decimal data. It will produce ambiguous results if used to manipulate non-decimal data. Particularly, if the four numeric bits of a character have a binary numeric value of 12 or more (octal 14, 15, 16, or 17), the character is treated as if it were a zero. The two remaining cases (octal 12 and 13) are unspecified.

EXAMPLE

Divide the 4-character integer whose leftmost location is location 1000 by the 3-character field whose rightmost location is location 500. Store the quotient in the leftmost locations of the field at 1000, and store the remainder in the rightmost locations of this field.

Na (number of characters in divisor) = 3

Ndd (number of characters in dividend) = 4

B (B address) = 1000

Units position of quotient (B-Na+Ndd-2) = 1000-3+4-2 = location 993

Units position of remainder (B+Ndd-1) = 1000+4-1 = location 1003

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1					D	500,1,000	
2							
3							

LOGIC INSTRUCTIONS

The following nine instructions are included in this category.

EXTRACT
HALF ADD
SUBSTITUTE
COMPARE
BRANCH
BRANCH ON CONDITION TEST
BRANCH ON CHARACTER CONDITION
BRANCH IF CHARACTER EQUAL
BRANCH ON BIT EQUAL

The Extract, Half Add, and Substitute instructions manipulate data on an individual basis, combining bits from two different fields according to rules based on AND/OR logic. Each of the remaining instructions causes a program branch to be performed either unconditionally (viz., the Branch instruction) or contingent upon the existence of a precisely defined condition.

EXT	EXTRACT (Logical Product)

FORMAT

	Op Code	Address Field	Address Field
a.	31	A	B
b.	31	A	
c.	31		

FUNCTION

Format a: The data in the A field is combined bit-by-bit with the data in the B field, according to the following rules. The result is stored in the B field.

BIT IN A FIELD	BIT IN B FIELD	BIT IN RESULT FIELD
1	1	1
1	0	0
0	1	0
0	0	0

Format b: The data in the A field is combined bit-by-bit with the data specified by (BAR), according to the rules stated above. The result is stored in the B field.

Format c: The data specified by (AAR) is combined with the data specified by (BAR), according to the rules stated above. The result is stored in the B field.

WORD MARKS

Formats a, b, and c:

A word mark is required for the shorter of the two operands. The operation terminates when this word mark is sensed.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-Nw	B-Nw
Format b:	NXT	A-Nw	Bp-Nw
Format c:	NXT	Ap-Nw	Bp-Nw

EXAMPLE

Remove all zone bits in the field tagged BASE by combining the contents of BASE with the contents of the field tagged CON. Each character in CON must have the following format:

Bit position B A 8 4 2 1

Contents 0 0 1 1 1 1

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ____ OF ____

	CARD NUMBER	TYPE	NAR	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1					EXT	CON, BASE	
2							
3							

HA	HALF ADD (Exclusive Or)

FORMAT

	Op Code	Address Field	Address Field
a.	30	A	B
b.	30	A	
c.	30		

FUNCTION

Format a: The data in the A field is combined with the data in the B field, according to the rules stated below. The result is stored in the B field.

BIT IN A FIELD	BIT IN B FIELD	BIT IN RESULT FIELD
1	1	0
1	0	1
0	1	1
0	0	0

Format b: The data in the A field is combined with the data specified by (BAR), according to the rules stated above. The result is stored in the B field.

Format c: The data specified by (AAR) is combined with the data specified by (BAR), according to the rules stated above. The result is stored in the B field.

WORD MARKS

Formats a, b, and c:

A word mark is required for the shorter of the two operands. The operation terminates when this word mark is sensed.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-Nw	B-Nw
Format b:	NXT	A-Nw	Bp-Nw
Format c:	NXT	Ap-Nw	Bp-Nw

EXAMPLE

Clear all the numeric bits in the field tagged SEVEN to zeros by combining (SEVEN) with (TOO). Do not change the zone bits in SEVEN. (The contents of each character in TOO are 00xxxx, where x equals the corresponding bit in SEVEN.)

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
				HA	TOO, SEVEN	

SST	SUBSTITUTE

FORMAT

	Op Code	Address Field	Address Field	Variant
a.	32	A	B	V
b.	32	A	B	
c.	32	A		
d.	32			

FUNCTION

Format a: The single character specified by the A address is compared with the variant character and is moved to the location specified by the B address, according to the following rules:

1. The A-character bit is transferred to the B address if the corresponding variant bit = 1.
2. The B-character bit is preserved if the corresponding variant bit = 0.

Format b: The single character specified by the A address is compared with the variant character specified in a previous instruction and is moved to the location specified by the B address, according to the rules stated above.

Format c: The single character specified by (A) is compared bit-by-bit with the variant character specified in a previous instruction and is moved to the location specified by (BAR), according to the rules stated above.

Format d: The single character specified by (AAR) is compared bit-by-bit with the variant character specified in a previous instruction and is moved to the location specified by (BAR), according to the rules stated above.

WORD MARKS

Formats a, b, and c:

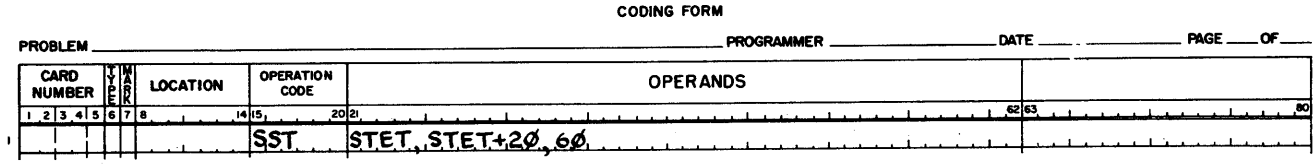
Word marks are not required in either field.

ADDRESS REGISTERS AFTER OPERATION

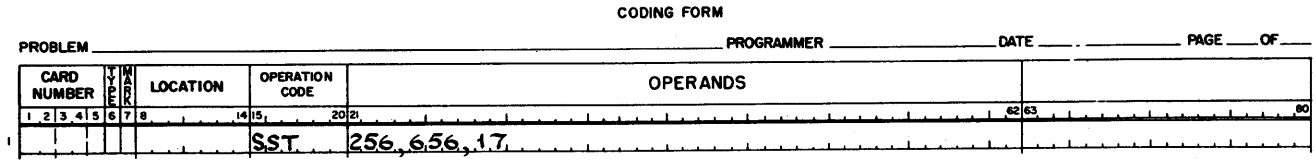
	SR	AAR	BAR
Format a:	NXT	A-1	B-1
Format b:	NXT	A-1	B-1
Format c:	NXT	A-1	Bp-1
Format d:	NXT	Ap-1	Bp-1

EXAMPLES

1. Move the zone bits from the location tagged STET to the location tagged STET + 20. A variant character of octal 60 provides the required variant bit configuration (i.e., 110 000).



2. Move the numeric portion of the character at location 256 to location 656. A variant of octal 17 provides the required variant bit configuration (i.e., 001 111).



C | COMPARE

FORMAT

	Op Code	Address Field	Address Field
a.	33	A	B
b.	33	A	
c.	33		

FUNCTION

Format a: The data in the B field is compared bit-by-bit with the data in the A field. The comparison turns on indicators that can be interrogated by subsequent Branch instructions. The indicators are reset by the next Compare instruction.

Format b: The data specified by (BAR) is compared bit-by-bit with the data in the A field. This operation turns on indicators which can be tested by subsequent Branch instructions. The indicators are reset by the next Compare instruction.

Format c: The data specified by (BAR) is compared bit-by-bit with the data in the field specified by (AAR). The comparison turns on indicators that can be interrogated by subsequent Branch instructions. The indicators are reset by the next Compare instruction.

WORD MARKS

Formats a, b, and c:

The word mark associated with the B operand terminates the operation. The A operand must have a word mark only if it is shorter than the B operand. In this case, transmission of data from the A field stops after the A-operand word mark is sensed, and the remaining characters of the B operand are compared with zeros. If the A operand is longer than the B operand, the characters of the A operand that exceed the field length defined by the B-operand word mark are not processed.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-Nw	B-Nb
Format b:	NXT	A-Nw	Bp-Nb
Format c:	NXT	Ap-Nw	Bp-Nb

NOTES

1. All characters that can appear in storage can be compared. The ascending order of characters is listed in Appendix D.
2. Both fields must have exactly the same bit configurations to be equal. For example, plus zero is not equal to minus zero.
3. Comparison results and associated branch conditions are listed below.

COMPARISON RESULT	BRANCH CONDITION
B<A	Low compare
B=A	Equal compare
B≤A	Low or equal compare
B>A	High compare
B≠A	Unequal compare
B≥A	High or equal compare

EXAMPLE

Compare item number with 4000. If item number equals 4000, continue the program in sequence; otherwise, branch to location NITEM.

Description	Tag
Item Number	ITEM
4000	CON4

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ . ___ PAGE ___ OF ___

	CARD NUMBER	TYPER	WARR	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1					C	CON4, ITEM	
2					BCT	NITEM, 45	
3							
4							

B BRANCH

FORMAT

a.

Op Code	Address Field	Address Field	Variant
65	A		

FUNCTION

The Branch instruction causes the program to branch to the location specified by the A address and to store the contents of the sequence register (SR) in the B-address register. It is used to interrupt normal program sequence and to continue the program at any desired point, without testing for specific conditions. Thus, this instruction is frequently referred to as an "unconditional branch."

WORD MARKS

Word marks are not affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

SR	AAR	BAR
JI(A)	A	NXT

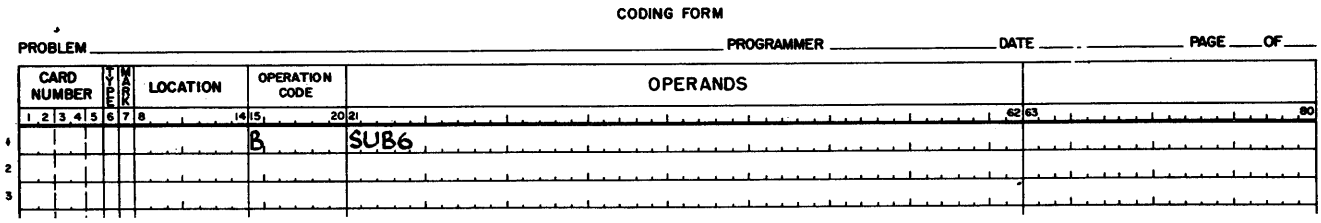
NOTES

1. The A address is placed in AAR during the extraction of this instruction, preserving any active high-order bits in AAR. When the instruction is executed, the entire contents of AAR specify the address to which the program branches. Also, the entire contents of SR are stored in BAR during the execution phase.

2. The contents of the variant register are unspecified following the execution of this instruction. Therefore, an instruction requiring a variant character must not be chained following a Branch instruction.

EXAMPLE

Select the next instruction from the location tagged SUB6.



BCT	BRANCH ON CONDITION TEST

FORMAT

	Op Code	Address Field	Address Field	Variant
a.	65	A		V
b.	65			

FUNCTION

Format a: The variant character specifies a condition indicator or a SENSE switch to be tested. If the condition being tested is present, the program branches to the location specified by the A address and the contents of SR are stored in the B-address register. If the condition specified by the variant character is not present, the program continues in sequence. Tables 6-5 and 6-6 list the valid variant characters and the conditions they test.

Format b: If the condition specified by the previous variant character is present, the program branches to the location specified by the (AAR) and the contents of SR are stored in BAR. If the condition being tested is not present, the program continues in sequence. Tables 6-5 and 6-6 list the valid variant characters and the conditions they test.

WORD MARKS

Formats a, b, and c:

Word marks are not affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR	
Format a:	JI(A)	A	NXT	BRANCH
	NXT	A	Bp	NO BRANCH
Format b:	JI(Ap)	Ap	NXT	BRANCH
	NXT	Ap	Bp	NO BRANCH

NOTES

1. If the overflow indicator is tested and an overflow condition exists, the indicator is automatically reset as a result of being tested. In all other cases, the indicator tested is not reset as a result of the test.
2. The comparison indicators are:
   a. set by the Compare instruction;
   b. stored (and cleared) by the Store Variant and Indicators instruction;
   c. restored by the Restore Variant and Indicators instruction;
   d. restored by the Resume Normal Mode instruction if coming out of the external interrupt mode (but not out of internal interrupt mode);
   e. stored when an external interrupt occurs.
3. The A address (if any) is placed in AAR during the extraction of this instruction, preserving any active high-order bits in AAR. If the instruction causes a branch (i.e., if the condition being tested is present), the entire contents of AAR specify the address to which the program branches when the instruction is executed. Also, the entire contents of SR are stored in BAR during the execution phase of the instruction.
4. Consider the variant character in its 6-bit form V6V5V4V3V2V1. The following chart may be used to determine the variant character to be used in a BCT instruction.

V_6	V_5	V_4	V_3	V_2	V_1
00 = Test SENSE Switches 1 through 4		SENSE Switch 4	SENSE Switch 3	SENSE Switch 2	SENSE Switch 1
01 = Test SENSE Switches 5 through 8		SENSE Switch 8	SENSE Switch 7	SENSE Switch 6	SENSE Switch 5
1 = Test Zero Balance, Overflow, or Compare	Zero Balance	Overflow	High Compare	Equal Compare	Low Compare

5. SENSE switches 5 through 8 are a standard feature with the VLF processor.

Table 6-5. SENSE Switch Test Conditions for BCT Instruction

Variant Character (Octal)	Branch On
00	Unconditional
01	SENSE Switch 1 On
02	SENSE Switch 2 On
03	SENSE Switches 1 and 2 On
04	SENSE Switch 3 ON
05	SENSE Switches 1 and 3 On
06	SENSE Switches 2 and 3 On
07	SENSE Switches 1, 2, and 3 On
10	SENSE Switch 4 On
11	SENSE Switches 1 and 4 On
12	SENSE Switches 2 and 4 On
13	SENSE Switches 1, 2, and 4 On
14	SENSE Switches 3 and 4 On
15	SENSE Switches 1, 3, and 4 On
16	SENSE Switches 2, 3, and 4 On
17	SENSE Switches 1, 2, 3, and 4 On
20	Unconditional
21	SENSE Switch 5 On
22	SENSE Switch 6 On
23	SENSE Switches 5 and 6 On
24	SENSE Switch 7 On
25	SENSE Switches 5 and 7 On
26	SENSE Switches 6 and 7 On
27	SENSE Switches 5, 6, and 7 On
30	SENSE Switch 8 On
31	SENSE Switches 5 and 8 On
32	SENSE Switches 6 and 8 On
33	SENSE Switches 5, 6, and 8 On
34	SENSE Switches 7 and 8 On
35	SENSE Switches 5, 7, and 8 On
36	SENSE Switches 6, 7, and 8 On
37	SENSE Switches 5, 6, 7, and 8 On

NOTE: When testing for a multiple SENSE switch condition, a branch occurs only if all of the specified conditions are met.

Table 6-6. Indicator Test Conditions for BCT Instruction

Variant Character (Octal)	Branch On
40	Do not branch
41	B<A (Low Compare
42	B=A (Equal Compare
43	B≤A (Low or Equal Compare)
44	B>A (High Compare)
45	B≠A (Unequal Compare)
46	B≥A (High or Equal Compare)
47	Unconditional
50	Overflow
51	Overflow or B<A
52	Overflow or B=A
53	Overflow or B≤A
54	Overflow or B>A
55	Overflow or B≠A
56	Overflow or B≥A
57	Unconditional
60	Zero Balance
61	Zero Balance or B<A
62	Zero Balance or B=A
63	Zero Balance or B≤A
64	Zero Balance or B>A
65	Zero Balance or B≠A
66	Zero Balance or B≥A
67	Unconditional
70	Overflow or Zero Balance
71	Overflow or Zero Balance or B<A
72	Overflow or Zero Balance or B=A
73	Overflow or Zero Balance or B≤A
74	Overflow or Zero Balance or B>A
75	Overflow or Zero Balance or B≠A
76	Overflow or Zero Balance or B≥A
77	Unconditional

NOTE: When testing for a multiple indicator condition, a branch occurs if any one of the specified conditions is met.

EXAMPLE

Subtract CREDIT from TOTAL and test for a zero balance. If this condition exists, branch to BZRO; otherwise, continue the program in sequence.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ . ___ PAGE ___ OF ___

CARD NUMBER 1 2 3 4 5	TYPE 6	MARK 7	LOCATION 8 ... 14	OPERATION CODE 15 ... 20	OPERANDS 21 ... 62	63 ... 80
1				S	CREDIT, TOTAL	
2				BCT	BZRO, 60	
3						
4						
5						

BCC | BRANCH ON CHARACTER CONDITION

FORMAT

	Op Code	Address Field	Address Field	Variant
a.	54	A	B	V
b.	54	A	B	
c.	54	A		
d.	54			

FUNCTION

Format a: The single character specified by the B address is examined for the condition specified by the variant character. If the condition is present, the program branches to the location specified by the A address, and the contents of SR are stored in the B-address register. If the condition is not present, the program continues in sequence. The valid variant characters and the condition each represents are listed in Tables 6-7 and 6-8.

Format b: The single character specified by the B address is examined for the condition specified by the variant character of a previous instruction. If the condition is present, the program branches to the location specified by the A address, and the contents of SR are stored in BAR. If the condition is not present, the program continues in sequence. The valid variant characters and the condition each represents are listed in Tables 6-7 and 6-8.

Format c: The single character specified by the contents of BAR is examined for a condition specified by the variant character of a previous instruction. If the condition is present, the program branches to the location specified by the A address, and the contents of SR are stored in BAR. If the condition is not present, the program continues in sequence. The valid variant characters and the condition each represents are listed in Tables 6-7 and 6-8.

Format d: The single character specified by (BAR) is examined for a condition specified by the variant character of a previous instruction. If the condition is present, the program branches to the location specified by (AAR), and contents of SR are stored in BAR. If the condition is not present, the program continues in sequence. The valid variant characters and the condition each represents are listed in Tables 6-7 and 6-8. The VLF processor, which is equipped with the advanced programming instruction can interpret any bit configuration of the variant character, ranging from octal 00 to octal 77. The valid variant characters which can be interpreted with this option are shown in Table 6-8.

Table 6-7. Basic Test Conditions for BCC Instruction

Variant Character (Octal)	Character Condition
00	Unconditional
02	The B bit of the character at B is 1.
06	The character at B contains a negative sign (the B and A bits are 10).
10	The character at B contains either a word mark or a record mark (the word-mark bit is 1).
12	The B bit is 1 and the word-mark bit is 1.
16	The character at B contains a negative sign and the word-mark bit is 1.
20	The character at B contains either an item mark or a record mark (the item-mark bit is 1).
22	The B bit is 1 and the item-mark bit is 1.
26	The character at B contains a negative sign and the item-mark bit is 1.
30	The character at B contains a record mark (the word-mark and item-mark bits are 11).
32	The character at B contains a record mark and the B bit is 1.
36	The character at B contains a record mark and a negative sign.

WORD MARKS

Formats a, b, c, and d:

Word marks are not affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR	
Format a:	JI(A)	A	NXT	BRANCH
	NXT	A	B-1	NO BRANCH
Format b:	JI(A)	A	NXT	BRANCH
	NXT	A	B-1	NO BRANCH
Format c:	JI(A)	A	NXT	BRANCH
	NXT	A	Bp-1	NO BRANCH
Format d:	JI(Ap)	Ap	NXT	BRANCH
	NXT	Ap	Bp-1	NO BRANCH

Table 6-8. BCC Test Conditions with Advanced Programming Instructions

Variant Character (Octal)	Character Condition
X0	No condition.
X1	The A bit of the character at B is 1.
X2	The B bit of the character at B is 1.
X3	The B and A bits of the character at B are 11.
X4	The B and A bits of the character at B are 00.
X5	The character at B contains a positive sign (the B and A bits are 01).
X6	The character at B contains a negative sign (the B and A bits are 10).
X7	The B and A bits of the character at B are 11 (same as X3 above).
0X	No condition.
1X	The word-mark bit of the character at B is 1 (either a word mark or a record mark is present).
2X	The item-mark bit of the character at B is 1 (either an item mark or a record mark is present).
3X	The character at B contains a record mark.
4X	The character at B contains no punctuation mark.
5X	The character at B contains a word mark.
6X	The character at B contains an item mark.
7X	This is a special case; see note 2.

NOTES:

1. An X represents any octal digit. If both octal digits specify "no condition" (i.e., 00), the branch occurs unconditionally. If only one digit is 0, the branch occurs if the condition specified by the other digit is met. If both octal digits specify conditions, the branch occurs if both conditions are met. The variant character 7X is an exception to these rules, as described in note 2.
2. The VLF processor interprets the 7X variant as follows:
   a. If X is 0, the branch is an unconditional branch.
   b. If X is any digit other than 0, the branch occurs if either the condition specified by the rightmost digit is met or the character at B contains a word mark.

NOTES

1. If the octal configuration of the variant character is 00 or 70, the branch is unconditional.

2. The A address (if any) is placed in AAR during the extraction of the BCC instruction, preserving any active high-order bits in AAR. If the instruction causes a branch (i. e., if the condition being tested is present), the entire contents of AAR specify the address to which the program branches when the instruction is executed. Also, the entire contents of SR are placed in BAR during the execution phase.

EXAMPLE

If the location tagged END contains a negative sign, branch to the location tagged NFIELD. Otherwise, continue the program in sequence.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ____ . ____ PAGE ____ OF ____

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 ... 14	15, ... 20	21 ... 62	63 ... 80
1					BCC	NFIELD, END, Ø6	
2							
3							

BCE | BRANCH IF CHARACTER EQUAL

FORMAT

	Op Code	Address Field	Address Field	Variant
a.	55	A	B	V
b.	55	A	B	
c.	55	A		
d.	55			

FUNCTION

Format a: The single character specified by the B address is compared to the variant character. If the bit configurations of the two characters are equal, the program branches to the location specified by the A address, and the contents of SR are stored in the B-address register. If the bit configurations are unequal, the program continues in sequence.

Format b: The single character specified by the B address is compared to the variant character specified in a previous instruction. If the bit configurations of the two characters are equal, the program branches to the location specified by the A address, and the contents of SR are stored in BAR. If the bit configurations are unequal, the program continues in sequence.

Format c: The single character specified by (BAR) is compared to the variant character specified in a previous instruction. If the bit configurations of the two characters are equal, the program branches to the location specified by the A address, and the contents of SR are stored in BAR. If the bit configurations are unequal, the program continues in sequence.

Format d: The single character specified by (BAR) is compared to the variant character specified in a previous instruction. If the bit configurations of the two characters are equal, the program branches to the location specified by (AAR), and the contents of SR are stored in BAR. If the bit configurations are unequal, the program continues in sequence.

WORD MARKS

Formats a, b, c, and d:

A word mark in the location tested has no effect on the instruction.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR	
Format a:	JI(A)	A	NXT	BRANCH
	NXT	A	B-1	NO BRANCH
Format b:	JI(A)	A	NXT	BRANCH
	NXT	A	B-1	NO BRANCH
Format c:	JI(A)	A	NXT	BRANCH
	NXT	A	Bp-1	NO BRANCH
Format d:	JI(Ap)	Ap	NXT	BRANCH
	NXT	Ap	Bp-1	NO BRANCH

NOTES

1. The A address (if any) is placed in AAR during the extraction of the BCE instruction, preserving any active high-order bits in AAR. If the instruction causes a branch (i.e., if the condition being tested is present), the entire contents of AAR specify the address to which the program branches when the instruction is executed. Also, the entire contents of SR are placed in BAR during the execution phase.

EXAMPLES

1. Determine if the character stored in the location tagged LABEL + 3 is equal to 6. If so, branch to the location tagged P6; otherwise, continue the program in sequence.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ . ___ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1				BCE	P6, LABEL+3, 6	
2						

2. Determine if any character position in the 7-character field tagged PART contains the letter Q. If so, branch to the location tagged RETRO; otherwise, continue the program in sequence.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ . ___ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1				BCE	RETRO,,PART,,Q	
2				BCE		
3				BCE		
4				BCE		
5				BCE		
6				BCE		
7				BCE		

BBE | BRANCH ON BIT EQUAL

FORMAT

	Op Code	Address Field	Address Field	Variant
a.	56	A	B	V
b.	56	A	B	
c.	56	A		
d.	56			

FUNCTION

Format a: The single character specified by the B address is combined bit-by-bit with the variant character, according to the rules shown below. If the result (the logical product) is not equal to zero, the program branches to the location specified by the A address, and the contents of SR are stored in the BAR. If the result is equal to zero, the program continues in sequence.

Bit in B Character	Bit in Variant Character	Bit in Result Field
1	1	1
1	0	0
0	1	0
0	0	0

Format b: The single character specified by the B address is combined bit-by-bit with the variant character specified in a previous instruction, according to the rules shown above. If the result is not equal to zero, the program branches to the location specified by the A address, and the contents of SR are stored in BAR. If the result is equal to zero, the program continues in sequence.

Format c: The single character specified by (BAR) is combined bit-by-bit with the variant character specified in a previous instruction, according to the rules shown above. If the result is not equal to zero, the program branches to the location specified by the A address, and the contents of SR are stored in BAR. If the result is equal to zero, the program continues in sequence.

Format d: The single character specified by (BAR) is combined bit-by-bit with the variant character specified in a previous instruction, according to the rules shown above. If the result is not equal to zero, the program branches to the location specified by the (AAR), and the contents of SR are stored in BAR. If the result is equal to zero, the program continues in sequence.

WORD MARKS

Formats a, b, c, and d:

Words marks are not tested by this instruction and have no effect on the operation.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR	
Format a:	JI(A)	A	NXT	BRANCH
	NXT	A	B-1	NO BRANCH
Format b:	JI(A)	A	NXT	BRANCH
	NXT	A	B-1	NO BRANCH
Format c:	JI(A)	A	NXT	BRANCH
	NXT	A	Bp-1	NO BRANCH
Format d:	JI(Ap)	Ap	NXT	BRANCH
	NXT	Ap	Bp-1	NO BRANCH

NOTES

1. The logical product formed by this instruction is tested, but is not stored. Main memory locations are not disturbed by this operation.
2. The A address (if present) is placed in AAR during the extraction of the instruction, preserving any active high-order bits in AAR. If the instruction causes a branch (i.e., if the logical product does not equal zero), the entire contents of AAR specify the address to which the program branches when the instruction is executed. Also, the entire contents of SR are placed in BAR during the execution phase.
3. Since this instruction results in a branch if any bit product is not equal to zero, only one bit at a time can be tested. Other bits can be checked by branching to additional BBE instructions.

EXAMPLE

Branch to the location tagged BIT8 only if the character at the location tagged MAR contains a 1 in both the B and the 8 bit positions. Otherwise, continue in sequence. This example requires two BBE instructions to test the two bits in question; location BIT8 is reached only if both tests are met.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ____ OF ____

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1					BBE	BITB,MAR,4Ø	
2					≀	≀	
3					≀	≀	
4				BITB	BBE	BITS,MAR,1Ø	
5				≀	≀	≀	
6				≀	≀	≀	
7				BITS	- - -	- - - - - -	

CONTROL INSTRUCTIONS

The following instructions are considered in this section.

SET WORD MARK
SET ITEM MARK
CLEAR WORD MARK
CLEAR ITEM MARK
HALT
NO OPERATION
MOVE CHARACTERS TO WORD MARK
LOAD CHARACTERS TO A-FIELD WORD MARK
STORE CONTROL REGISTERS
LOAD CONTROL REGISTERS
CHANGE ADDRESSING MODE
CHANGE SEQUENCING MODE
EXTENDED MOVE
MOVE AND TRANSLATE
MOVE ITEM AND TRANSLATE
LOAD INDEX/BARRICADE INDICATOR
STORE INDEX/BARRICADE INDICATOR
TABLE LOOKUP

The majority of the instructions in this category are used to manipulate data within control memory, to prepare main memory storage areas for the processing of data fields, and to control the sequential selection and interpretation of instructions in the stored program.

Some of these instructions are used to move data within the main memory. Control over the placement of punctuation can be exercised, and data can be translated to other forms as it is moved.

SW | SET WORD MARK

FORMAT

	Op Code	Address Field	Address Field
a.	22	A	B
b.	22	A	
c.	22		

FUNCTION

Format a: A word mark is set at the location specified by each address. The data and item-mark bits at each location are undisturbed.

Format b: A word mark is set at the location specified by the A address. The data and item-mark bits at this location are undisturbed.

Format c: Word marks are set at the locations specified by the contents of the A- and B-address registers (AAR and BAR). The data and item-mark bits at each location are undisturbed.

WORD MARKS

Formats a, b, and c:

Word marks are set as described above.

ADDRESS REGISTERS AFTER OPERATION

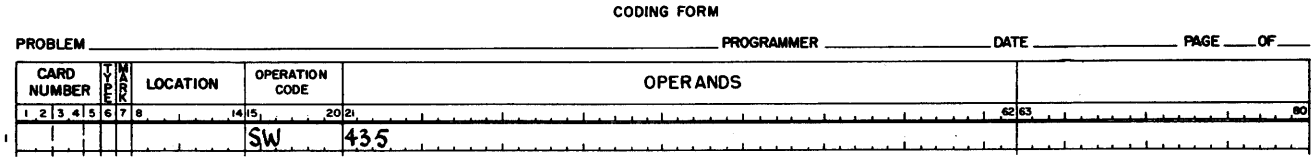
	SR	AAR	BAR
Format a:	NXT	A-1	B-1
Format b:	NXT	A-1	A-1
Format c:	NXT	Ap-1	Bp-1

NOTE

The extraction of this instruction when coded in format a. automatically terminates when the last character of the B address is loaded into BAR. Therefore, a word mark is not required in the location following the B address.

EXAMPLE

Set a word mark in location 435.



SI	SET ITEM MARK

FORMAT

	Op Code	Address Field	Address Field
a.	20	A	B
b.	20	A	
c.	20		

FUNCTION

Format a: An item mark is set at the location specified by each address. The data and word-mark bits at each location are undisturbed.

Format b: An item mark is set at the location specified by the A address. The data and word-mark bits at this location are undisturbed.

Format c: Item marks are set at the locations specified by the contents of the A and B address registers. The data and word-mark bits at each location are undisturbed.

WORD MARKS

Formats a, b, and c:

Word marks are not affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-1	B-1
Format b:	NXT	A-1	A-1
Format c:	NXT	Ap-1	Bp-1

NOTE

The extraction of this instruction when coded in format a. automatically terminates when the last character of the B address is loaded into BAR. Therefore, a word mark is not required in the location following the B address.

EXAMPLE

Set item marks in the locations tagged ENT and ENT+80.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
				SI	ENT, ENT+80	

CW	CLEAR WORD MARK

FORMAT

	Op Code	Address Field	Address Field
a.	23	A	B
b.	23	A	
c.	23		

FUNCTION

Format a: The locations specified by the A and B addresses are cleared of word marks. The data and item-mark bits at these locations are undisturbed.

Format b: The word mark at the location specified by the A address is cleared. The data and item-mark bits at this location are undisturbed.

Format c: Word marks are cleared at the locations specified by (AAR) and (BAR). The data and item-mark bits at these locations are undisturbed.

WORD MARKS

Formats a, b, and c:

Word marks are cleared as defined above.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-1	B-1
Format b:	NXT	A-1	A-1
Format c:	NXT	Ap-1	Bp-1

EXAMPLE

Clear the word marks at locations 400 and 435.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ . ___ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62 / 63 ... 80
1				CW	400,435
2					
3					

CI CLEAR ITEM MARK

FORMAT

	Op Code	Address Field	Address Field
a.	21	A	B
b.	21	A	
c.	21		

FUNCTION

Format a: Item marks are cleared from the locations specified in the A and B addresses. The data and word-mark bits at these locations are undisturbed.

Format b: The item mark at the location specified by the A address is cleared. The data and word-mark bits at this location are undisturbed.

Format c: Item marks are cleared at the locations specified by the (AAR) and (BAR). The data and word-mark bits at these locations are undisturbed.

WORD MARKS

Formats a, b, and c:

Word marks are not affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-1	B-1
Format b:	NXT	A-1	A-1
Format c:	NXT	Ap-1	Bp-1

EXAMPLE

Clear the item mark in location REC.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1					CI	REC	
2							
3							
4							

H	HALT

FORMAT

	Op Code	Address Field	Address Field	Variant 1
a.	45			
b.	45	A		
c.	45	A	B	
d.	45	A	B	V

FUNCTION

Format a: This instruction causes the machine to stop. Pressing the RUN button causes the program to resume with the next instruction in sequence.

Format b: The contents of SR are stored in the B-address register; the A address of the instruction is transferred to SR; then the machine stops. Pressing the RUN button causes the program to resume with the instruction specified in the A address. This format is usually referred to as a "halt and branch" instruction.

Format c: This instruction causes the machine to stop. Pressing the RUN button causes the program to resume with the next instruction in sequence. The address portions can be used to indicate control information such as a halt identification number (see note 2).

Format d: This instruction causes the machine to stop. Pressing the RUN button causes the program to resume with the next instruction in sequence. The address portions and the variant character can be used to indicate control information such as halt identification number (see note 2).

WORD MARKS

Formats a, b, c, and d:

Word marks are not affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	Ap	Bp
Format b:	JI(A)	A	NXT
Format c:	NXT	A	B
Format d:	NXT	A	B

NOTES

1. If a Halt instruction (in any format) is executed during a peripheral transfer, the transfer continues until it is completed.
2. Formats c. and d. are useful when a program contains a number of halts. By assigning a number or symbol in the A and B addresses to each halt, the programmer can later identify a particular halt by displaying the contents of AAR and/or BAR. Although the contents of the variant register cannot be displayed through the console, format d. can be used to store a variant character which can subsequently be used by the program.
3. The Halt op code is a "privileged" op code that has special significance in the VLF processor when storage protection is in effect (see Appendix F).

EXAMPLES

1. Stop the machine and specify that when the RUN button is pressed, the next instruction will be selected from the location tagged RES.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ . ___ PAGE ___ OF ___

CARD NUMBER	TYPE	VARR	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
				H	RES	

2. Identify the halt at the end of a job as follows:

A address = 9
B address = 9

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ . ___ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1				H	9,9	
2						
3						

NOP	NO OPERATION

FORMAT

a.

Op Code	Address Field	Address Field
40		

FUNCTION

This instruction performs no operation. This op code can be substituted for the op code of any instruction to make that instruction ineffective.

WORD MARKS

Program operation resumes at the next op code identified by a word mark.

ADDRESS REGISTERS AFTER OPERATION

SR	AAR	BAR
NXT	Ap	Bp

NOTES

1. This instruction is commonly used in program modification to cause the machine to skip over specific instructions.
2. Information appearing in an address portion of an instruction for which the NOP instruction is substituted is not loaded into the associated operand address register. The final character of such information, however, is loaded into the variant register.

EXAMPLE

Reserve one storage location for an operation code such as Branch. When the op code B is inserted, the NOP instruction will be modified to branch to location SWX.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ . ___ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1				NOP	SWX	
2						

MCW	MOVE CHARACTERS TO WORD MARK

FORMAT

	Op Code	Address Field	Address Field
a.	14	A	B
b.	14	A	
c.	14		

FUNCTION

Format a: The data and item-mark bits in the A field are moved to the B field.

Format b: The data and item-mark bits in the A field are moved to the field specified by the (BAR).

Format c: The data and item-mark bits in the field specified by (AAR) are moved to the field specified by (BAR).

WORD MARKS

Formats a, b, and c:

A word mark (or record mark) is required in the shorter of the two fields. The operation terminates when this mark is sensed.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-Nw	B-Nw
Format b:	NXT	A-Nw	Bp-Nw
Format c:	NXT	Ap-Nw	Bp-Nw

NOTE

Item marks initially stored in B-field locations will be cleared if the corresponding A-field characters do not include item marks.

EXAMPLE

Move the following A fields and store them in sequential B fields as shown.

Description	A Field	B Field
Unit Number	150-155	800-805
Rack Number	160-168	806-814
Part Number	173-180	815-822
Pin Number	185-187	823-825

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ____ OF ____

CARD NUMBER	TYPE	NAR	LOCATION	OPERATION CODE	OPERANDS	
1				MCW	187,825	
2				MCW	180	
3				MCW	168	
4				MCW	155	
5						
6						

LCA	LOAD CHARACTERS TO A-FIELD WORD MARK

FORMAT

	Op Code	Address Field	Address Field
a.	15	A	B
b.	15	A	
c.	15		

FUNCTION

Format a: The data and punctuation bits in the A field are transferred to the B field.

Format b: The data and punctuation bits in the A field are transferred to the field specified by the (BAR).

Format c: The data and punctuation bits in the field specified by the (AAR) are transferred to the field specified by (BAR).

WORD MARKS

Formats a, b, and c:

The A operand must have a defining word mark (or record mark). The operation terminates when this mark is transferred to the B field.

ADDRESS REGISTERS AFTER OPERATION

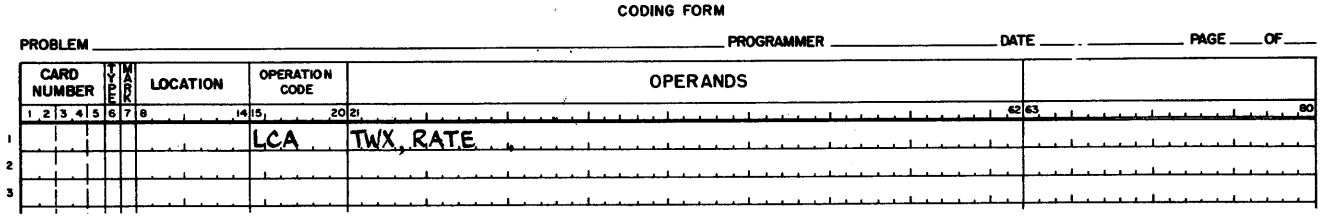
	SR	AAR	BAR
Format a:	NXT	A-Na	B-Na
Format b:	NXT	A-Na	Bp-Na
Format c:	NXT	Ap-Na	Bp-Na

NOTES

1. This instruction (in any format) is the only instruction that always moves both a field and its defining punctuation mark.
2. All punctuation (word marks, item marks, and record marks) initially stored in B-field locations will be cleared if the corresponding A-field characters do not include identical punctuation.
3. The B address must never fall within the A field. The A address may fall within the B field, however, if desired.

EXAMPLE

Move both the data bits and the defining word mark of the field tagged TWX to the field tagged RATE.



SCR	STORE CONTROL REGISTERS

FORMAT

	Op Code	Address Field	Address Field	Variant
a.	24	A		V
b.	24	A		
c.	24			

FUNCTION

Format a: The contents of the control memory register specified by the variant character are stored in the field whose units position is defined by the A address of this instruction. The method of storing these contents depends on the addressing mode being used, as shown in Table 6-9. The valid variant characters and the control register each character represents are listed in Table 6-10.

Format b: The contents of the control memory register specified by the variant character in a previous instruction are stored in the field whose units position is defined by the A address of this instruction. The number of bits stored depends on the addressing mode being used, as shown in Table 6-9. The valid variant characters and the control register each represents are listed in Table 6-10.

Format c: The contents of the control memory register specified by the variant character in a previous instruction are stored in the field whose units position is defined by the contents of the AAR. The number of bits stored depends on the addressing mode being used, as shown in Table 6-9. The valid variant characters and the control register each character represents are listed in Table 6-10.

Table 6-9. Control Register Contents Stored by SCR Instruction

Addressing Mode	Amount of Control Register Stored
Two-Character	Low-order two characters (12 bits).
Three-Character	Low-order 15 bits; the high-order three bits of the field specified by the A address are cleared to zeros.
Four-Character	The entire contents of the control register plus sufficient high-order zeros to make up 24 bits.
NOTE: All bit positions not required to address the largest memory address in a user's system are set to zeros in the A field.	

Table 6-10. Control Registers Stored by SCR Instruction

Variant Character (Octal)	Control Register	Variant Character (Octal)	Control Register	Variant Character (Octal)	Control Register
00	CLC7	20	CLC8	64	CSR
01	CLC1	21	CLC4	66	EIR
02	CLC2	22	CLC5	67	AAR
03	CLC3	23	CLC6	70	BAR
04	CLC7'	24	CLC8'	76	IIR
05	CLC1'	25	CLC4'	77	SR
06	CLC2'	26	CLC5'		
07	CLC3'	27	CLC6'		
10	SLC7	30	SLC8		
11	SLC1	31	SLC4		
12	SLC2	32	SLC5		
13	SLC3	33	SLC6		
14	SLC7'	34	SLC8'		
15	SLC1'	35	SLC4'		
16	SLC2'	36	SLC5'		
17	SLC3'	37	SLC6'		

WORD MARKS

Formats a, b, and c:

A-operand punctuation neither affects nor is affected by the instruction.

ADDRESS REGISTERS AFTER OPERATION

Formats a, b, and c:

SR	AAR	BAR
NXT	Ap	Bp

NOTES

1. If AAR is specified by the variant character (octal 67), the previous address in AAR (not the A address retrieved from this instruction) is stored in the location specified by the A address.
2. The control memory register actually designated by the variant character 67_8 is a word register (not AAR). During the extraction of an SCR or LCR instruction, AAR is used to refer to the main memory. Prior to this, the previous contents of AAR are stored in the work register; at the end of the instruction, the contents of the work register are restored in AAR.

EXAMPLE

Store the contents of BAR in the A address of the Branch instruction tagged EXIT. (The processor is assumed to be in the 3-character addressing mode.)

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ____ . _____ PAGE ____ OF ____

CARD NUMBER	TYPER	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
			SUB	SCR	EXIT+3, 7Ø	
			EXIT	B	Ø	

LCR	LOAD CONTROL REGISTERS

FORMAT

	Op Code	Address Field	Address Field	Variant
a.	25	A		V
b.	25	A		
c.	25			

FUNCTION

Format a: The contents of the field specified by the A address are loaded into the control register specified by the variant character. The contents of the A field is another main memory address. The method of loading this

address into the specified control register depends on the addressing mode being used, as shown in Table 6-11. Variant characters and their associated control registers are the same as those specified for the Store Control Registers instruction (see Table 6-10).

Table 6-11. Control Register Contents Loaded by LCR Instruction

Addressing Mode	Amount of Memory Address Loaded
Two-Character	Two-character (12-bit) address is loaded into the low-order two character locations of the register. All other bits in the register (if any) are not disturbed (i.e., the bank bits are protected).
Three-Character	15-bit address is loaded into the low-order 15-bit locations of the register. All other bits in the register (if any) are not disturbed (i.e., the sector bits are protected).
Four-Character	An address up to 19 bits long is loaded into the register.

Format b: The contents of the field specified by the A address are loaded into the control register specified by the variant character in a previous instruction. The method of loading the contents of this field (another main memory address) depends on the addressing mode being used, as shown in Table 6-11. Variant characters and their associated control registers are the same as those specified for the Store Control Registers instruction.

Format c: The main memory address specified by (AAR) is loaded into the control register specified in a previous instruction. The method of loading this address into the specified register depends on the addressing mode being used, as shown in Table 6-11. Variant characters and their associated control registers are the same as those specified for the Store Control Registers instruction.

WORD MARKS

Formats a, b, and c:

A-operand punctuation neither affects nor is affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

Formats a, b, and c:

SR	AAR	BAR	
NXT	(A)	Bp	Variant = 67_8
NXT	Ap	(A)	Variant = 70_8
(A)	Ap	Bp	Variant = 77_8
NXT	Ap	Bp	All Others

NOTES

1. If SR is specified by the variant character (77_8), the next instruction is selected from the location whose address is stored in the field specified by the A address of the Load Control Registers instruction. In all other cases, the program continues in sequence.
2. The LCR op code is a "privileged op code that has special significance in the VLF processor when storage protection is in effect (see Appendix F).

EXAMPLE

Load the address stored in the location tagged SUB1 into the change sequence register (CSR).

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ . ___ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1				LCR	SUB1, 64	
2						
3						

CAM	CHANGE ADDRESSING MODE

FORMAT

	Op Code	Address Field	Address Field	Variant
a.	42			V
b.	42			

FUNCTION

Format a: The Change Addressing Mode instruction is used to specify the following conditions are designated by the variant character.

1. The addressing mode (2-, 3-, or 4-character) in which the processor is to interpret the address portions of all subsequent instructions (see note 1).
2. The processing mode (standard mode or "trap" mode) in which all subsequent instructions are to be processed. (See note 3 for a description of the trap mode.)

The variant characters and the mode(s) each character represents are listed in Table 6-12.

Format b: The variant character in a previous instruction specifies the addressing mode and processing mode in which all subsequent instructions are to be processed. The variant characters and the mode(s) each character represents are listed in Table 6-12.

Table 6-12. Modes Specified by Variant Character in CAM Instruction

Variant Character (Octal)	Mode(s)
20	Two-character, standard mode
00 or 40	Three-character, standard mode
60	Four-character, standard mode
24	Two-character, trap mode
04 or 44	Three-character, trap mode
64	Four-character, trap mode

WORD MARKS

Formats a and b:

Word marks are not affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

Formats a and b:

SR	AAR	BAR
NXT	Ap	Bp

NOTES

1. The CAM instruction is used in conjunction with the ADMODE assembly control statement to specify addressing mode. The ADMODE statement directs the assembly program to assemble the address portions of all subsequent source program instructions as 2-, 3-, or 4-character addresses. The CAM instruction directs the processor to interpret the address portions of all subsequent object program instructions as 2-, 3-, or 4-character addresses. Thus, an address assembled in the 3-character addressing mode (via an ADMODE statement) must be processed during a program run as a 3-character address for proper execution; the processor is placed in the 3-character addressing mode during object program execution by the CAM instruction.

2. The ability to change addressing modes within a program makes it possible to save both time and memory space and provides greater programming flexibility. Extraction and execution time is saved when a smaller addressing mode is used, due to the elimination of the extra memory cycles necessary for a larger address (in characters). Memory space may be conserved by storing frequently used subroutines in the 2-character addressing mode (see example).

   The larger addresses are necessary to address larger continuous portions of memory. For instance, a 2-character address can specify only memory locations within a 4,096-character bank of memory. A 3-character address can refer to any location in a 32,768-character sector. A 4-character address can directly address any location in the entire memory (from location 0_{10} to location $524,288_{10}$).

3. When the processor is in the trap mode of instruction execution, any instruction whose op code contains an item mark (or record mark) is both extracted and executed as if it were a Change Sequencing Mode instruction (see page 6-70) regardless of the op code that is actually present. The A address, B address, and variant character (if any) of the instruction are delivered to AAR, BAR, and the variant register, respectively. The "trapped" op code is not executed; a Change Sequencing Mode instruction (CSM) is executed in its place. The CSM instruction causes a branch to the location stored in the change sequence register (CSR); this location is the beginning of a routine to interpret and execute the instruction whose op code was trapped.

EXAMPLE

Figure 6-9 shows the coding which provides entry to and exit from a subroutine to be executed in the 2-character addressing mode. Both an ADMODE statement and a CAM instruction must be coded (in either order) at the beginning and end of the subroutine. However, only the CAM instructions are stored in the main memory. (Since CAM instructions have no address portions, the manner in which they are stored is not affected by an ADMODE statement.)

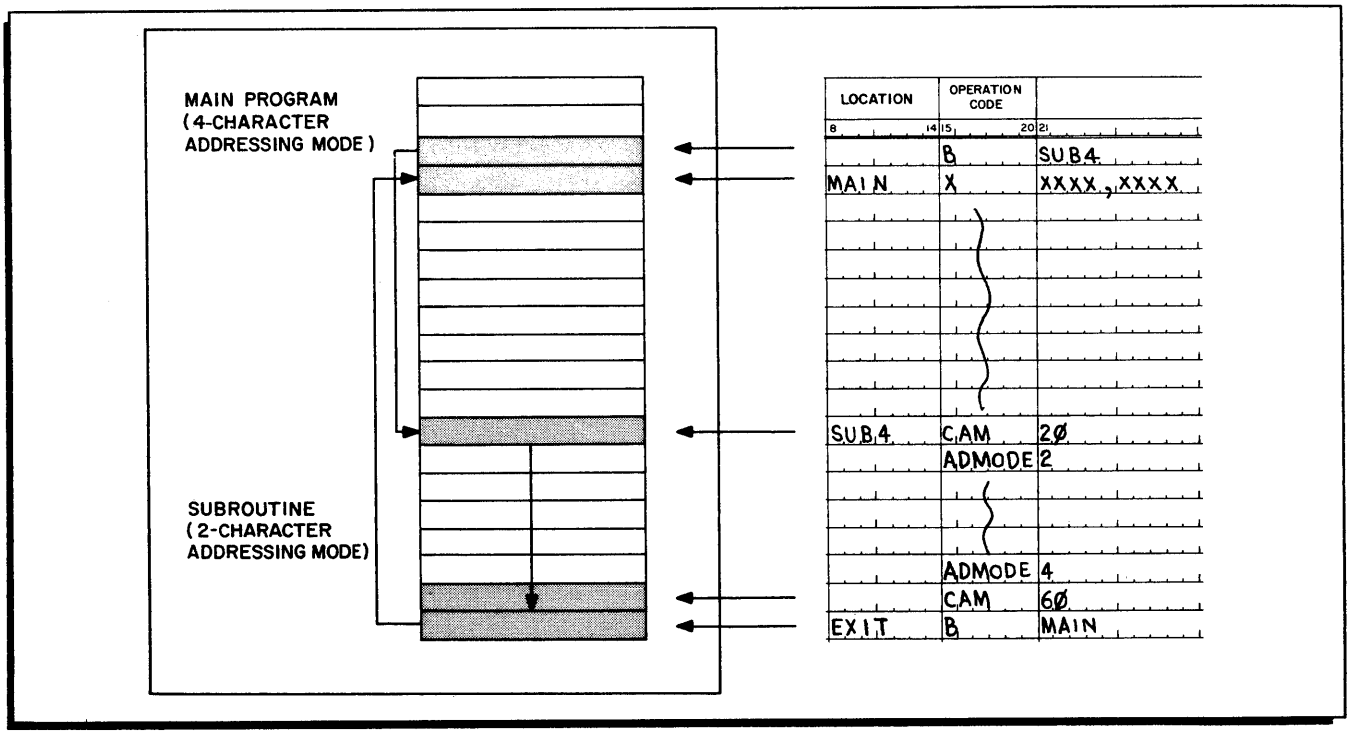


Figure 6-9. Changing Addressing Modes via CAM Instruction

NOTE: The branch from the main program to SUB4 in Figure 6-9 could have been caused by an item-marked op code (if the processor were in the trap mode) instead of by the Branch instruction. In this case, the memory location tagged SUB4 would be stored in CSR, so that when the item-marked op code was encountered, the contents of SR and CSR would be interchanged. The program would automatically branch to SUB4 in this case.

CSM	CHANGE SEQUENCING MODE

FORMAT

	Op Code	Address Field	Address Field	Variant
a.	43			
b.	43	A		
c.	43	A	B	
d.	43	A	B	V

FUNCTION

Format a: The contents of the sequence register (SR) and the change sequence register (CSR) are interchanged, and the program branches to the address which was previously stored in CSR.

Format b: The contents of SR and CSR are interchanged, and the program branches to the address which was previously stored in CSR. The A address is loaded into the A-address register (AAR).

Format c: The contents of SR and CSR are interchanged, and the program branches to the address which was previously stored in CSR. The A and B addresses are loaded into AAR and BAR, respectively.

Format d: The contents of SR and CSR are interchanged, and the program branches to the address which was previously stored in CSR. The A and B addresses and the variant character are loaded into AAR, BAR, and the variant register, respectively.

WORD MARKS

Formats a, b, c, and d:

Word marks are not affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

	SR	CSR	AAR	BAR
Format a:	JI (contents of CSR)	NXT	Ap	Bp
Format b:	JI (contents of CSR)	NXT	A	Bp
Format c:	JI (contents of CSR)	NXT	A	B
Format d:	JI (contents of CSR)	NXT	A	B

NOTES

1. The Load Control Registers instruction can be used to set up the contents of CSR.
2. When the "trap" mode of instruction execution is specified by the Change Addressing Mode instruction, any subsequent instruction whose op code contains an item mark or a record mark is retrieved and executed as if it were a Change Sequencing Mode instruction.

EXAMPLE

Store the absolute address tagged CHANGE in CSR via a Load Control Registers instruction. Later, alter the program sequence by branching to the instruction tagged CHANGE. Provide for the ultimate return to normal programming sequence by storing the contents of SR in CSR.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

CARD NUMBER	TYPE	MAR K	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1				LCR	CHANGE,64	
2						
3						
4						
5				CSM		

EXM	EXTENDED MOVE

FORMAT

	Op Code	Address Field	Address Field	Variant
a.	10	A	B	V
b.	10	A	B	
c.	10	A		
d.	10			

FUNCTION

Format a: The contents of the A field are moved to the B field in the manner specified by the variant character (see Table 6-13). The programmer specifies how the move operation is to be performed by selecting the desired conditions from the table and encoding the resulting two octal digits as the variant character of the instruction.

Format b: The contents of the A field are moved to the B field in the manner specified by the variant character of a previous instruction (see Table 6-13).

Format c: The contents of the A field are moved to the field specified by (BAR) in the manner specified by the variant character of a previous instruction (see Table 6-13).

Format d: The contents of the field specified by (AAR) are moved to the field specified by (BAR) in the manner specified by the variant character of a previous instruction (see Table 6-13).

PUNCTUATION MARKS

Formats a, b, c, and d:

The A field must have a defining punctuation mark, except when the variant character specifies a 1-character transfer.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR	
Format a:	NXT	A-Na	B-Na	VARIANT = (0,2,4, or 6)X
	NXT	A+Na	B+Na	VARIANT = (1,3,5, or 7)X
Format b:	NXT	A-Na	B-Na	VARIANT = (0,2,4, or 6)X
	NXT	A+Na	B+Na	VARIANT = (1,3,5, or 7)X
Format c:	NXT	A-Na	Bp-Na	VARIANT = (0,2,4, or 6)X
	NXT	A+Na	Bp+Na	VARIANT = (1,3,5, or 7)X
Format d:	NXT	Ap-Na	Bp-Na	VARIANT = (0,2,4, or 6)X
	NXT	Ap+Na	Bp+Na	VARIANT = (1,3,5, or 7)X

Table 6-13. Extended Move Conditions

Variant Character (Octal)	Condition
X1	Move A-field data bits to corresponding bit positions in B field.
X2	Move A-field word-mark bits to corresponding bit positions in B field.
X3	Move A-field data and word-mark bits to corresponding bit positions in B field.
X4	Move A-field item-mark bits to corresponding bit positions in B field.
X5	Move A-field data and item-mark bits to corresponding bit positions in B field.
X6	Move A-field word-mark and item-mark bits to corresponding bit positions in B field.
X7	Move A-field data, word-mark and item-mark bits to corresponding bit positions in B field.
0X	Move one character from A to B. The A- and B-address registers are decremented by one.
1X	Move one character from A to B. The A- and B-address registers are incremented by one.

Table 6-13 (cont). Extended Move Conditions

Variant Character (Octal)	Condition
2X	Move characters from right to left (A and B addresses specify rightmost characters in operand fields). Terminate the operation when the first A-field word mark is sensed.
3X	Move characters from left to right (A and B addresses specify leftmost characters in operand fields). Terminate the operation when the first A-field word mark is sensed.
4X	Move characters from right to left. Terminate the operation when the first A-field item mark is sensed.
5X	Move characters from left to right. Terminate the operation when the first A-field item mark is sensed.
6X	Move characters from right to left. Terminate the operation when the first A-field record mark is sensed.
7X	Move characters from left to right. Terminate the operation when the first A-field record mark is sensed.

NOTES

1. Here is an example of a typical variant bit configuration: V = 110011. This configuration, encoded in octal notation as 63, specifies that A field data and word-mark bits are to be moved to the B field from right to left until the first record mark is sensed in the A field.
2. Consider the variant character in its 6-bit form, V6V5V4V3V2V1. If V1 = 0, A-operand data bits are not transferred and data bits in the B field remain unchanged.
3. If V2 = 0, A-operand word-mark bits are not transferred and B-operand word-mark bits remain unchanged.
4. If V3 = 0, A-operand item-mark bits are not transferred and B-operand item-mark bits remain unchanged.
5. The character containing the terminating punctuation is moved in the same manner as the rest of the field.

EXAMPLES

1. Move the data bits of the single character in the location 26 beyond that tagged TEMP to the location tagged WORK, and decrement the A- and B-address registers.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62 63 ... 80
1				EXM	TEMP+26, WORK, Ø1
2					
3					

2. Move only the data bits in the field tagged RESV to the field tagged WORK. Move the data from right to left, and terminate the operation when the first item mark in the A field is sensed.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62 63 ... 80
1				EXM	RESV, WORK, 41
2					
3					

MAT | MOVE AND TRANSLATE

FORMAT

	Op Code	Address Field	Address Field	Variant 1	Variant 2
a.	60	A	B	V	V

	Op Code	Address Field	Address Field	Address Field
b.	60	A	B	C

FUNCTION

Format a: The MAT instruction translates characters from one 6-bit configuration to another by means of a stored "translation table." The instruction can be used to translate any number of consecutive characters in the memory.

The A address specifies the location of the rightmost character in the field to be translated. The B address specifies the location into which the translated equivalent of the rightmost A-field character will be moved.

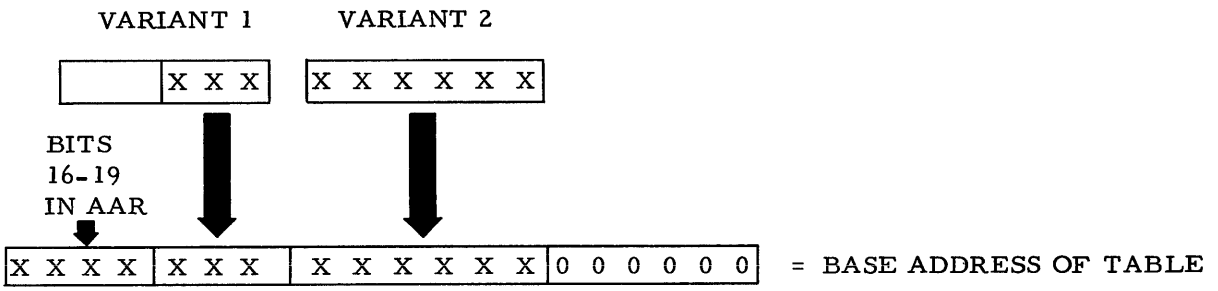
The operation normally terminates when an A-field word mark is sensed. The operation is also terminated if a character is transferred from a word-marked location within the translation table.

The address within the translation table which contains the translated equivalent of an A-field character is formed by combining the A-field character with the two variant characters. The method of combining these three characters depends on the addressing mode being used, as described below.

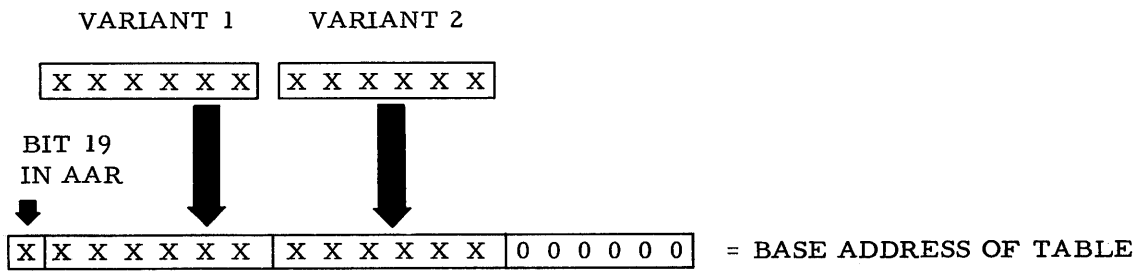
The leftmost, or base, address of the translation table is formed by combining variants 1, 2, and a zero character. If the processor is in 2- or 3-character addressing mode, the leftmost three bits of variant 1 are ignored and the corresponding bit positions (i.e., the sector bits)

in the base address (bits 16, 17, 18, and 19) are taken from the contents of the AAR. If the processor is in the 4-character addressing mode, the entire 6-bit contents of variant 1 form bits 13 through 18 of the base address, while the leftmost (nineteenth) bit, if present, is taken from the contents of AAR.

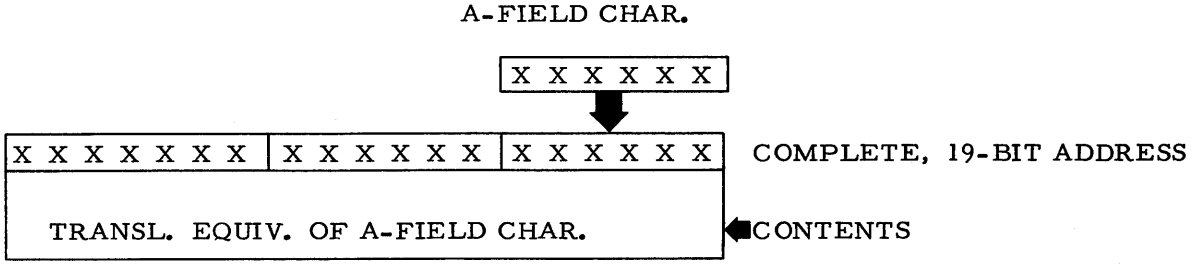
Two- or Three-Character Addressing Mode



Four-Character Addressing Mode



A character in the A field is translated when it is appended to the variant characters (in place of the zero character) to form a complete, 19-bit address. This complete address contains the translated equivalent of the appended A-field character (see below).



Note that because of the positions of variant 1 and variant 2 in the complete address, the base address of the table will always be a multiple of 64. This is compatible with translation requirements since each A-field character can have any of 64 bit configurations (see note 5).

It is a simple task to store the desired equivalent values in a translation table. For instance, assume that a character set which is to be translated into Honeywell code represents the letter A by the bit configuration 110001. Since this bit configuration represents a binary value of 49, the desired Honeywell equivalent (i. e., 010001) should be stored 49 locations beyond the base address of the translation table.

Format b: This is an alternate and simpler format for coding the MAT instruction. In this format, a "C address" replaces the variant characters used in format a. to define the base address of the table. Thus, format b. relieves the programmer of dealing with modulo-64 addresses and converting to octal notation each time a MAT instruction is coded.

The C address is a symbolic tag that is contained in the location field of another source-program entry (e. g. , a RESV statement). Once the absolute base address of the table is defined as described for format a., the C address is equated to that address and used in its stead whenever a MAT instruction using the same table is coded again in the program.

Example 2 shows how a C address can be used to define the base address of the translation table.

WORD MARKS

Formats a and b:

The A field must have a defining word mark. It is this word mark that normally stops the operation. The operation will also be terminated if a character is transferred from a word-marked location within the translation table.

ADDRESS REGISTERS AFTER OPERATION

Formats a and b:

SR	AAR	BAR
NXT	A-Nct	B-Nct

NOTES

1. This instruction cannot be chained.
2. The contents of the variant register following a move and translate operation are unspecified. Therefore, an instruction requiring a variant character must not be chained after an MAT instruction.
3. Item-mark bits as well as data bits are transferred from the translation table to the B field.
4. Word marks initially stored in the B field remain unchanged. They do not affect the execution of this instruction.
5. The base address of the translation table must always be a multiple of 64. The assembly program automatically stores the table in this manner when directed by a MORG assembly control statement containing an operand of 64.

EXAMPLES

1. Figure 6-10 shows how A-field data is moved to the B field via a translation table.

Translate the contents of the field tagged EXCODE using the stored translation table whose base address is 256_{10} (=400_8). Store the translated equivalent in the field tagged EQUIV.

A Address:	EXCODE (absolute value = location 630)
B Address:	EQUIV (absolute value = location 900)
Variant 1:	00 = base address of table (location 256)
Variant 2:	04

CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1				MAT	EXCODE, EQUIV, ØØ, Ø4	

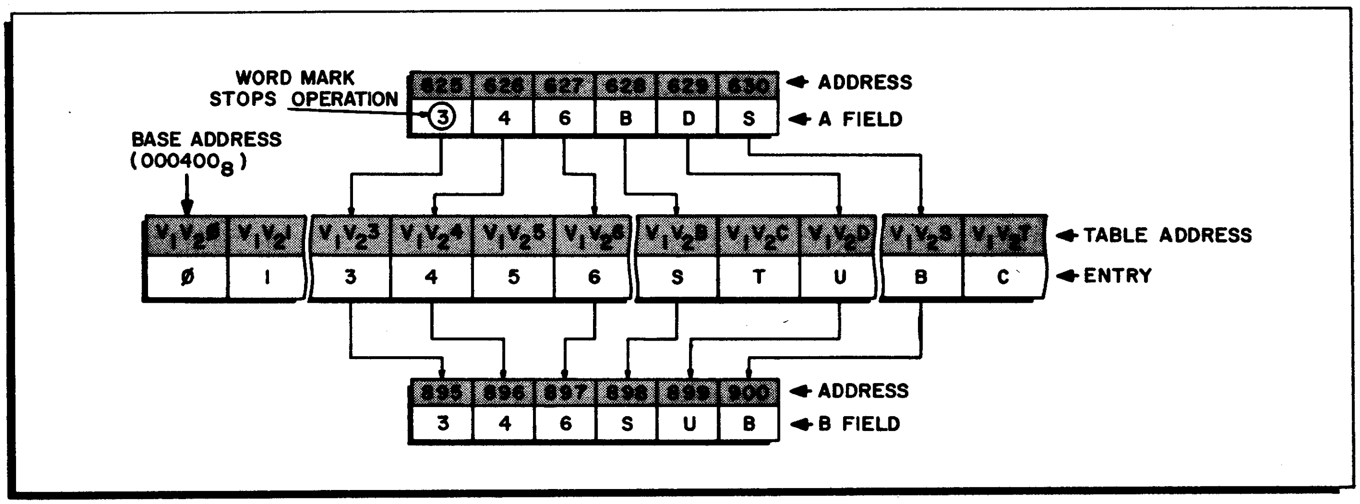


Figure 6-10. MAT Operation

2. The following coding shows how the preceding MAT instruction can be coded using a C address. The translation table is set up with a base address of 256_{10} by means of an ORG statement and two DC statements. The ORG statement directs the assembly program to load subsequent coding into memory locations beginning at location 256_{10}. The first DC statement defines an alphanumeric constant 40 characters long (i.e., the maximum size of an alphanumeric constant). These characters are the first 40 characters of a 64-character translation table. The second DC statement defines the remaining 24 characters of the table.

   When the MAT instruction is executed, the absolute address equated to the tag MATAB1 is used as the table's base address as in example 1.

CODING FORM

PROBLEM ______ PROGRAMMER ______ DATE ______ PAGE ___ OF ___

CARD NUMBER (1-5)	TYPE (6)	MARK (7)	LOCATION (8-14)	OPERATION CODE (15-20)	OPERANDS (21-62)	(63-80)
1				ORG	256	
2			MATAB1	DC	*Δ1234567890[X-8-2][0-8-2]@[8-6]Δ[0-8-7]/STUVWXYZΔ,[0-8-5]#[0-8-2][X-8-3]-JKLMNOP*	
3				DC	@QRŌ$*□[R-8-2]ΔABCDEFGHIΔ.[4-8-6]%[R-8-5]Δ@	
4				{		
5				MAT	EXCODE, EQUIV, MATAB1	

MIT	MOVE ITEM AND TRANSLATE

FORMAT

	Op Code	Address Field	Address Field	Variant 1	Variant 2	Variant 3
a.	62	A	B	V	V	V

	Op Code	Address Field	Address Field	Address Field	Variant
b.	62	A	B	C	V

FUNCTION

Format a: The Move Item and Translate instruction is used to translate any information unit (up to 12-bit code) to another information unit of up to 12 bits (e. g. , to Series 200 6-bit character code) by the use of a stored translation table. Any number of consecutive information units stored in the memory can be translated.

The A address is the leftmost address of the item to be translated. The B address is the leftmost address of the item into which the translated equivalent of the A item will be moved. The MIT instruction translates the data contents in the A item and moves the translated results, left to right, to the B item.

The operation normally terminates when an item mark is sensed in the A item. The operation will also be terminated if a word-marked character is encountered in the translation table.

An information unit up to six bits in length is stored in one 6-bit character location in the memory. Any information unit greater than six bits (7 through 12 bits) is stored in two successive 6-bit character locations. Thus, an information unit consisting of up to six bits is considered as a 6-bit character, and a unit of from 7 to 12 bits is considered as a "12-bit character."

The sizes of the information units involved in the operation are specified by variant 3, as shown in Table 6-14.

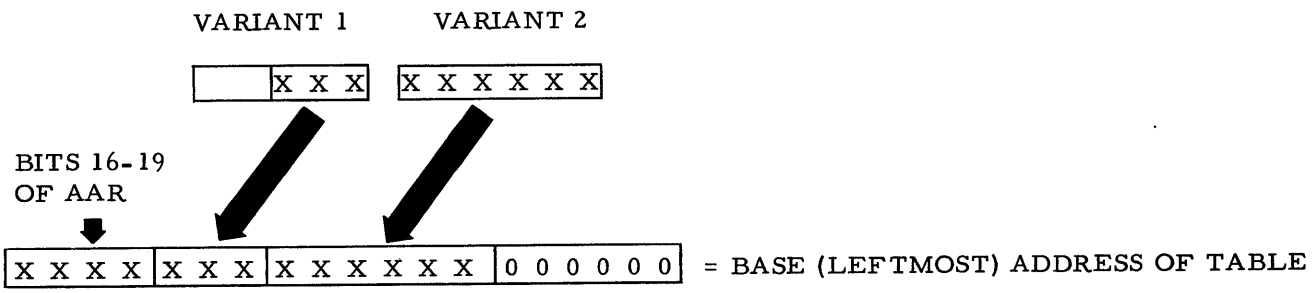
Table 6-14. Size of Information Units in MIT Operation

VARIANT 3	OPERATION
00	Translate each 6-bit character in the A item. Move the translated equivalent to a 6-bit character location in the B item.
01	Translate each 12-bit character in the A item. Move the translated equivalent to a 6-bit character location in the B item.
02	Translate each 6-bit character in the A item. Move the translated equivalent to two character locations (12 bits) in the B item.
03	Translate each 12-bit character in the A item. Move the translated equivalent to two character locations (12 bits) in the B item.

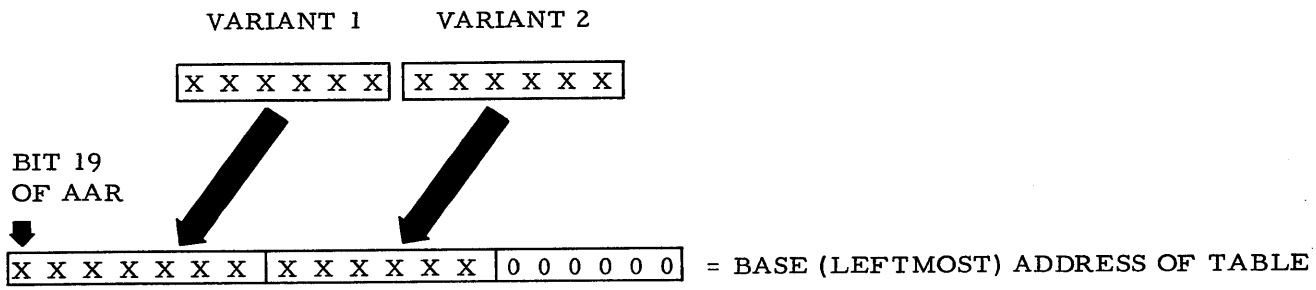
The desired equivalent of an A-item information unit is taken from the stored translation table and moved to the B item. Thus, if the desired equivalent is a 6-bit character, each table entry occupies one 6-bit character location in the table. If the desired equivalent is a 12-bit character, each table entry occupies two consecutive 6-bit character locations in the table. Consequently, variant 3 implicitly specifies the size of each table entry when it explicitly specifies the size of the B-item information unit.

The leftmost, or base, address of the translation table is formed by combining variants 1, 2, and a zero character as shown below. If the processor is in the 2- or 3-character addressing mode, the leftmost three bits of variant 1 are ignored and the corresponding bit positions (i.e., the sector bits) in the base address of the table are taken from the contents of the A-address register (AAR). If the processor is in the 4-character addressing mode, the entire 6-bit contents of variant 1 form bits 13 through 18 of the base address, and the nineteenth bit, if present, is taken from the contents of AAR.

Two- or Three-Character Addressing Mode



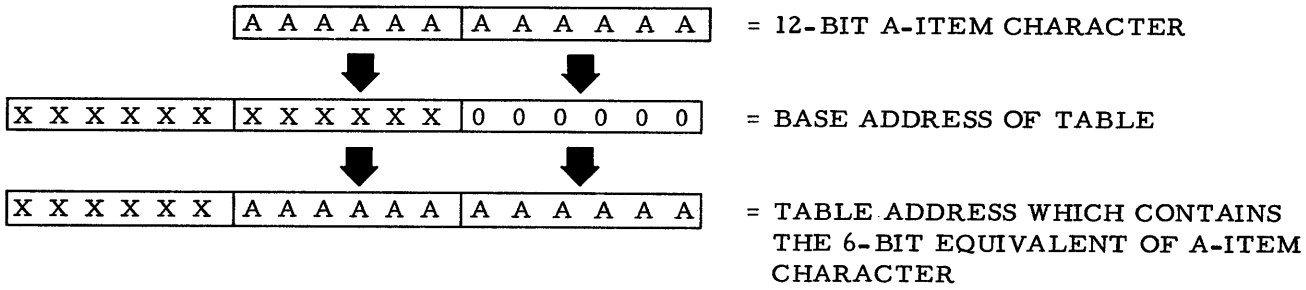
Four-Character Addressing Mode



The address within the translation table which contains the translated equivalent of an A-item character (6- or 12-bit) is formed by superimposing the A-item character over the base address of the table. The method of superposition depends on the size of each table entry (whether 6 or 12 bits), as described below. [1]

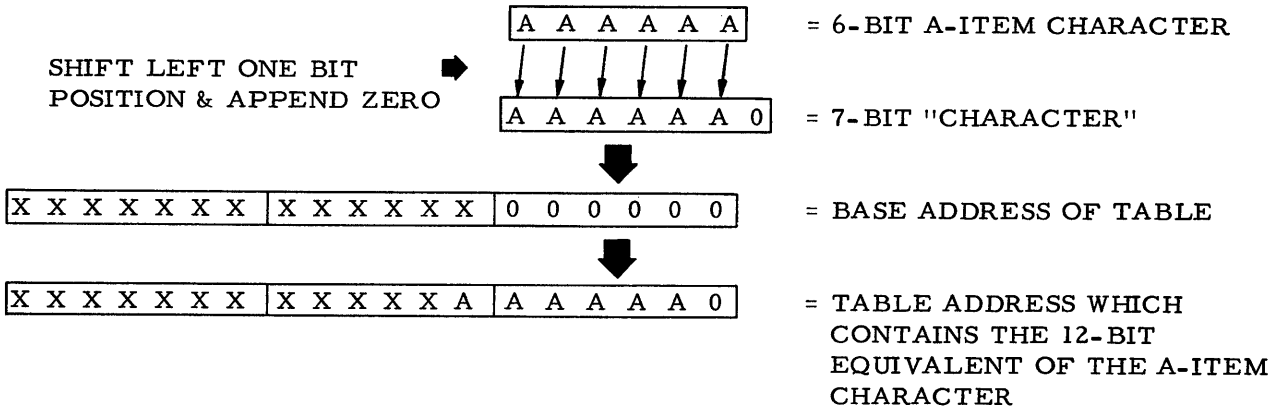
[1] Superposition is performed by placing a 1 bit in every position of the table address in which a 1 existed in either the A-item character or the base address or both. This is the "logical inclusive OR" function.

If each table entry is a 6-bit character (variant 3 = 00 or 01), the 6- or 12-bit A-item character is superimposed over the rightmost bit positions of the base address. The illustration below shows a 12-bit A-item character being superimposed over the base address, where A = an A-item bit and x = a base address bit.



If each table entry is a 12-bit character (variant 3 = 02 or 03), the 6- or 12-bit A-item character is first shifted one bit position to the left, forming a 7- or 13-bit "character." The rightmost bit position of the character is set to zero. The "character" is then superimposed over the base address to form the table address of the translated equivalent of the A-item character. The shift operation is used to double the referenced table address, since each table entry is stored in two, rather than one, 6-bit character locations. The resultant address is the address of the leftmost of the two successive 6-bit character locations in the table.

The illustration below shows how a 6-bit A-item character is shifted one bit position to the left and then superimposed over the translation table's base address to form the table address of its equivalent; A = an A-item bit, and X = a base address bit.



Format b: This is an alternate format for coding the MIT instruction. As in the MAT instruction, a symbolic tag replaces the variant characters used to define the base address of the table in format a. The tag is contained in the location field of another source-program entry which equates the tag to the base address of the table.

The second example of coding a MAT instruction shows the method by which a translation table is stored in memory so that the leftmost location of the table can be used as a symbolic address. This is identical to the method used for format b. of the MIT instruction.

PUNCTUATION MARKS

Formats a and b:

The A item must contain an item mark. It is this punctuation mark that normally stops the operation. If the A-item information units are 12-bit characters, the terminating item mark may appear in either of the two 6-bit character locations.

The operation will also be terminated if a character (6- or 12-bit) is encountered in a word-marked location in the translation table. In this case, neither the word-marked character nor any subsequent characters are moved to the B item; instead, a sequence change is performed (see note 5).

ADDRESS REGISTERS AFTER OPERATION

Formats a and b:

SR	CSR	AAR	BAR	
NXT	CSR_p	A+(NAu) (Nut)	B+(NBu) (Nut)	ITEM MARK IN A ITEM STOPS OPERATION
JI (contents) of CSR)	NXT	A+(NAu) (Nut)	B+(NBu) (Nut)	WORD MARK IN TABLE STOPS OPERATION

NOTES

1. This instruction cannot be chained.
2. The last 6-bit character referenced in the translation table (whether word-marked or not) is left in the variant register following the move item and translate operation.
3. Item-mark bits as well as data bits are transferred from the translation table to the B item.
4. Word marks initially stored in the B item remain unchanged. They do not affect the execution of this instruction.
5. A data control character (e. g., a case-shift character in a teletype code), rather than a translated equivalent to be transferred to the B item, can be stored in a word-marked location in the table. When this word-marked location is sensed, the character in that location is not moved; rather, the contents of SR and CSR are interchanged, providing entry to the routine whose beginning address was previously stored in the CSR. Since the word-marked character is stored in the variant register (see note 2), that character can be stored by a Store Variant and Indicators instruction and subsequently tested for identification in the routine.

6. The base address of the translation table must be a multiple of at least 64, due to the positions of variants 1 and 2 in the total 19-bit address. This requirement is sufficient only for the translation of 6-bit to 6-bit codes. If other than 6-bit codes are involved in the translation, the base address of the table must be a multiple of X (where X is the product of the number of codes defined by active bits in the A-field entries times the number of characters in each table entry). In other words, the base address of the table must be a multiple of the table size itself. The MORG assembly control statement can be used to assign memory locations to the translation table, starting with the next available memory location whose address is a multiple of 64, 128, 256, etc., as determined by the size of the table.

7. This instruction is a standard feature on the VLF processor.

EXAMPLE

Figure 6-11 shows how 8-bit code is translated to Series 200 6-bit character code by means of a stored translation table. Each 8-bit information unit is stored in two consecutive 6-bit character locations in the A-item tagged EIGHT.

Translate the data contents of the item tagged EIGHT using the translation table whose base address is location 512_{10} (1000_8). Store the translated values (6-bit characters) in the item tagged SIX.

A Address: EIGHT (absolute value = location 800)
B Address: SIX (absolute value = location 650)
Variant 1: 00 =
Variant 2: 10 = the address of table (location 512)
Variant 3: 01 =

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
				MIT	EIGHT,SIX,ØØ,1Ø,Ø1	

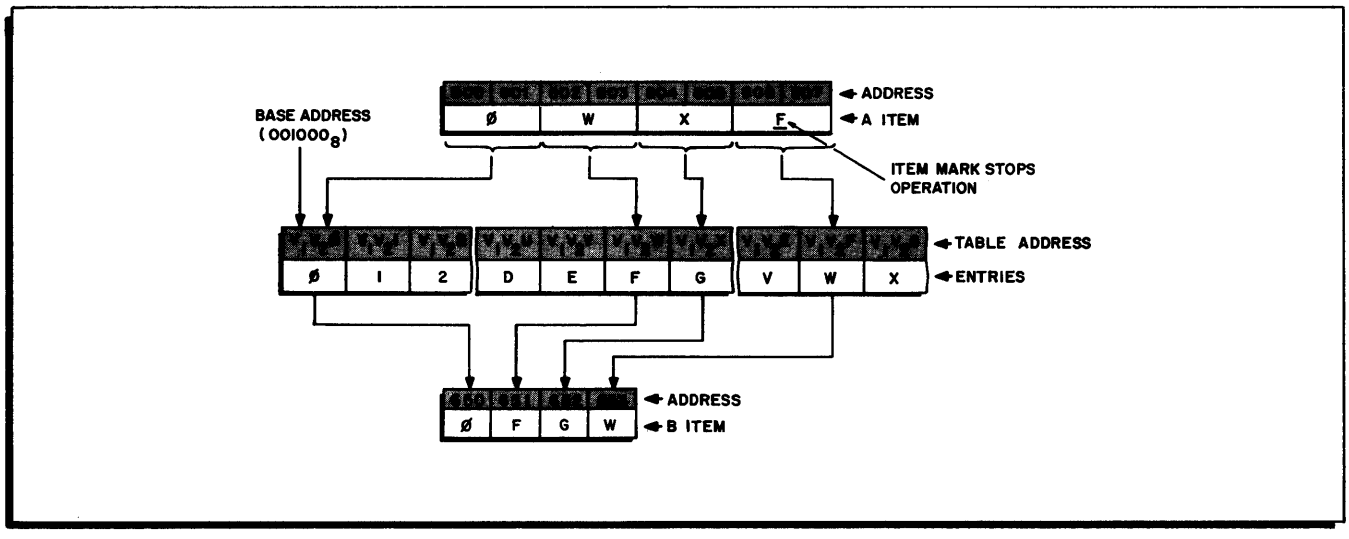


Figure 6-11. MIT Operation

LIB	LOAD INDEX/BARRICADE REGISTER

FORMAT

	Op Code	Address Field	Address Field
a.	77	A	
b.	77	A	B

FUNCTION

Format a: Basic storage protection is provided by this instruction format; the character(s) at the location(s) specified by the A address is loaded into the index/barricade register (IBR), specifying the number of a 4,096-character main memory bank. The leftmost location of the specified bank is the leftmost location of the protected memory area. (The rightmost location of the protected area is the rightmost location of memory.) A seventh bit and eighth bit, the rightmost bits of the contents of A-1, are loaded into IBR. Examples of the correspondence between the number loaded into IBR and the position of the barricade are shown in Table 6-15.

Format b: Storage protection with base relocation is provided by this instruction format; the index/barricade register (IBR) is loaded in the same manner as for basic storage protection (format a above), but the barricade is relocated relative to the base relocation address. Consequently, when storage protection is in effect, data cannot be delivered to memory locations above the barricade or below the base relocation address unless processing is in the interrupt mode. The character(s) at the location(s) specified by the B address is loaded into the base relocation register (BRR), specifying the integral multiple of a 4,096-character main memory bank. The number of main memory locations so designated augments all memory references made in the standard (noninterrupt) mode. A seventh bit and eighth bit, the rightmost bits of the contents of B-1, are loaded into BRR. The character(s) specified by the A address is loaded into the index/barricade register (IBR), specifying the integral multiple of a 4,096-character main memory bank. The barricade is established to the left of the leftmost location in the specified bank (as augmented by the base relocation address). A seventh bit and eighth bit, the rightmost bits of the contents of A-1, are loaded into IBR. Examples of the correspondence between the number loaded into IBR and the position of the barricade are shown in Table 6-15.

WORD MARKS

Formats a and b:

Word marks are not affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-2	Bp
Format b:	NXT	A-2	B-2

NOTES

1. Fifteen additional index registers are located in the leftmost 60 character locations of the main memory bank specified by this instruction. These locations can be used as normal storage locations when they are not being used for indexing operations.

2. The LIB op code is a "privileged" op code that has special significance in the VLF processor when storage protection is in effect (see Appendix F).

Table 6-15. Leftmost Boundaries of Protected Memory

Contents of A or of A+(A-1)		Leftmost Boundary of Protected Memory	Contents of A or of A+(A-1)		Leftmost Boundary of Protected Memory
Octal	Decimal		Octal	Decimal	
00	0	0	200	128	524, 288
01	1	4, 096	210	136	557, 056
10	8	32, 768	220	144	589, 824
20	16	65, 536	230	152	622, 592
30	24	98, 304	240	160	655, 360
40	32	131, 072	250	168	688, 128
50	40	163, 840	260	176	720, 896
60	48	196, 608	270	184	753, 664
70	56	229, 376	300	192	786, 432
100	64	262, 144	310	200	819, 200
110	72	294, 912	320	208	851, 968
120	80	327, 680	330	216	884, 736
130	88	360, 448	340	224	917, 504
140	96	393, 216	350	232	950, 272
150	104	425, 984	360	240	983, 040
160	112	458, 752	370	248	1, 015, 808
170	120	491, 520	377	255	1, 044, 480

EXAMPLE

Assuming that there are 131,072 storage locations in the main memory, set up the memory in such a way that the "open" memory area consists of locations 0 through 65,535 and the protected memory area consists of locations 65,536 through 131,072. The single octal character "20" is contained in the location tagged MP2.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ . ___ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1				LIB	MP2	
2						
3						

SIB | STORE INDEX/BARRICADE REGISTER

FORMAT

	Op Code	Address Field	Address Field
a.	76	A	
b.	76	A	B

FUNCTION

Format a: Basic storage protection is provided by this instruction format; the contents (up to seven bits) of the index/barricade register (IBR) are stored in the character location(s) specified by the A address. All high-order bit positions in A which are not used to specify the contents of the index/barricade register are cleared to zeros. A seventh and eighth bit in IBR are stored in the rightmost bit positions of location A-1 and the four remaining bit positions in A-1 are cleared to zeros.

Format b: Storage protection with base relocation is provided by this instruction format; the contents of the index/barricade register (IBR) are stored in the same manner as for basic storage protection (format a, above); in addition, the contents of the base relocation register (BRR) are also stored. The contents (up to eight bits) of BRR are stored in the character location(s) specified by the B address. All high-order bit positions in B which are not used to specify the contents of the base relocation register are cleared to zeros. A seventh and eighth bit in BRR are stored in the rightmost bit positions of location B-1 and the four remaining bit positions in B-1 are cleared to zeros. The contents (up to eight bits) of the index/barricade register are stored in the character location(s) specified by the A address.

All high-order bit positions in A which are not used to specify the contents of the index/barricade register are cleared to zeros. A seventh and eighth bit in IBR are stored in the rightmost bit positions of location A-1 and the four remaining bit positions in A-1 are cleared to zeros.

WORD MARKS

Formats a and b:

Word marks are not affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-2	Bp
Format b:	NXT	A-2	B-2

EXAMPLE

Store the contents of the index/barricade register in the single character location tagged PROT.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1				SIB	PROT	
2						
3						

TLU	TABLE LOOKUP

FORMAT

	Op Code	Address Field	Address Field	Variant
a.	57	A	B	V
b.	57	A	B	
c.	57	A		
d.	57			

FUNCTION

Format a: A table in memory is a series of fields, each of which normally contains an argument of a function and the corresponding value of the function (see notes 1 and 2). The Table Lookup instruction initiates a search in a stored table for an argument which bears a specified relationship to a search argument, which is stated in the instruction.

The B address specifies the rightmost location of the stored table, the A address specifies the location of the search argument, and the variant character specifies a relationship (equal to, higher than, etc.) between the desired argument in the table and the search argument. The table is searched from right to left until this relationship is found or until a table field is found which is shorter than the search argument. Then, comparison indicators are turned on and the search terminates.

Format b: Search the table whose rightmost location is specified by B for an argument which bears to the search argument specified by A a relationship specified by the variant character of a previous instruction. When this relationship is found or when a table field is found which is shorter than the search argument, turn on comparison indicators and terminate the search.

Format c: Search the table whose rightmost location is specified by the contents of the B-address register for an argument which bears to the search argument specified by A a relationship specified by the variant character of a previous instruction. When this relationship is found or when a table field is found which is shorter than the search argument, turn on comparison indicators and terminate the search.

Format d: Search the table whose rightmost location is specified by (BAR) for an argument which bears to the search argument specified by the (AAR) a relationship specified by the variant character of a previous instruction. When this relationship is found or when a table field is found which is shorter than the search argument, turn on comparison indicators and terminate the search.

WORD MARKS

Formats a, b, c, and d:

The A operand (the search argument) must have a defining word mark. Each table field must also have a defining word mark.

ADDRESS REGISTERS AFTER OPERATION

	SR	AAR	BAR
Format a:	NXT	A-Na	Lta
Format b:	NXT	A-Na	Lta
Format c:	NXT	A-Na	Lta
Format d:	NXT	Ap-Na	Lta

NOTES

1. Each value in the table is normally stored immediately to the left of the corresponding argument, and each pair (argument plus value) constitutes a field in the table. However, if the values in the table are longer than three characters, it is advisable to store them in another part of memory and to store their 2- or 3-character addresses in the table instead. Since the timing of the TLU instruction depends on the number of characters searched in the table, it is desirable to minimize the length of the table.

2. The Branch on Condition Test instruction can be used after the Table Lookup instruction to branch to a routine which moves the located value to a work area. Note that at the completion of the TLU instruction, the B-address register contains the address of the desired value (or the address of a location containing the address of the desired value, in the case where the values are too long for efficient storage in the table).

3. The variant characters which specify the desired relationships between the search argument and the argument to be located in the table are as follows:

Variant Character (Octal)	Relationship Which Terminates Search
01	Stored Argument < Search Argument
02	Stored Argument = Search Argument
03	Stored Argument ≤ Search Argument
04	Stored Argument Search Argument
05	Stored Argument ≠ Search Argument
06	Stored Argument ≥ Search Argument

4. The length of each argument in the table must be equal to the length of the search argument. Note that a short table field (e.g., one which contains a short argument or which contains no value) can be used to terminate the search, which leaves the comparison indicators set to the condition "Stored Argument" > Search Argument."

EXAMPLE

1. Figure 6-12 shows now a stored table is searched for an argument which bears a specified relationship to a search argument.

   Search the table tagged TABLE1 for the value which corresponds to the argument (557) stored in the field tagged ARGMNT.

   A Address: ARGMNT (absolute value = location 609)
   B Address: TABLE1 (absolute value = location 149)
   Variant 1: 02 = (Stored Argument = Search Argument)

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ____ PAGE ____ OF ____

CARD NUMBER	TYPE	VAR	LOCATION	OPERATION CODE	OPERANDS
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62
				TLU	ARGMNT, TABLE1, Ø2

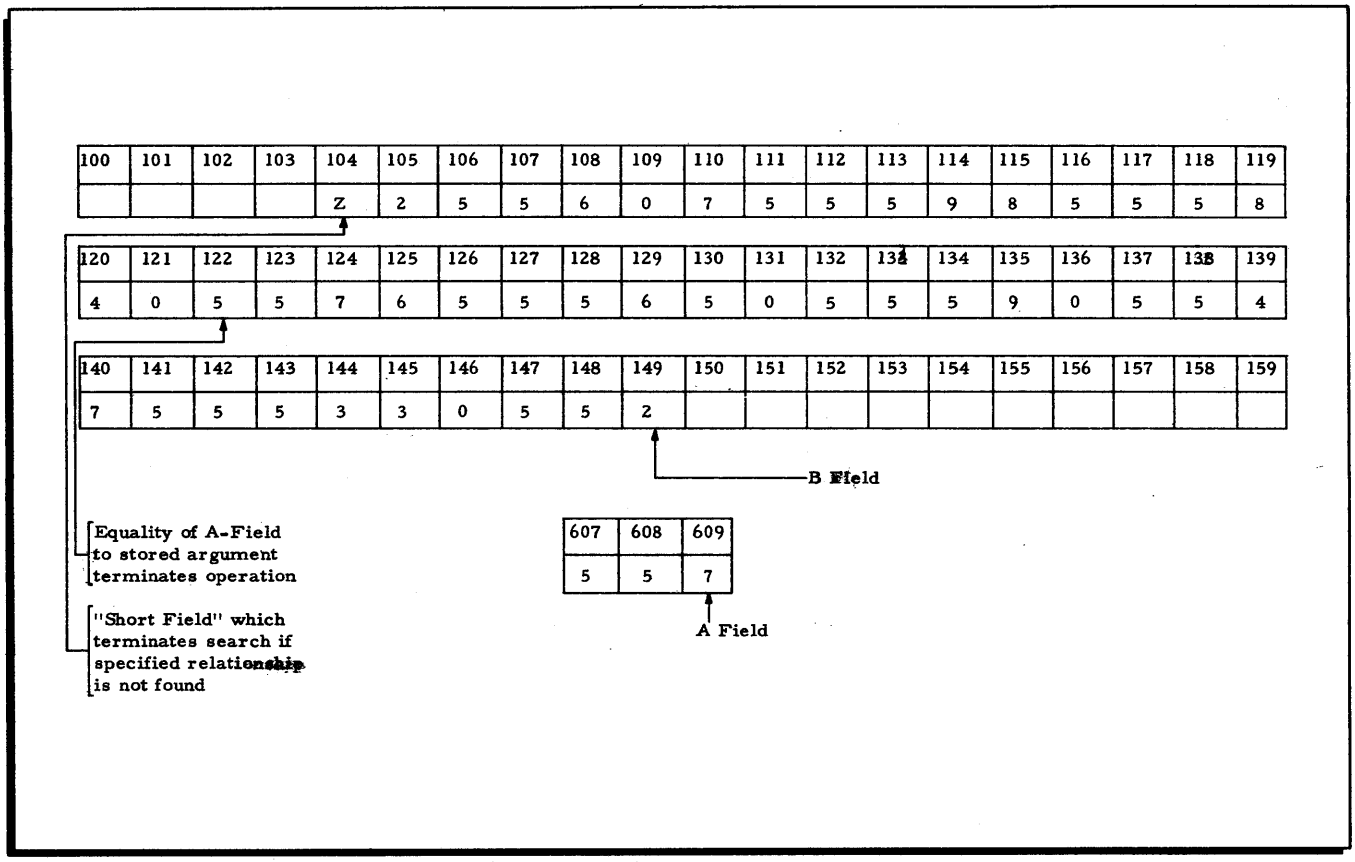


Figure 6-12. TLU Operation

INTERRUPT CONTROL INSTRUCTIONS

The following instructions are included in this category.

STORE VARIANT AND INDICATORS
RESTORE VARIANT AND INDICATORS
MONITOR CALL
RESUME NORMAL MODE

The normal processing sequence can be interrupted by any one of four sources: (1) a peripheral control; (2) the operator's console; (3) an internal processor condition which violates a protected portion of memory; or (4) a programmed instruction. Program control automatically branches to a stored routine which identifies the source of interruption and services the condition which caused the interruption.

Four instructions are used in conjunction with the automatic interrupt facility. The Monitor Call instruction is one of the four interrupt sources. The remaining three instructions, coded in the interrupt routine, help identify the source, store information which will be needed when the normal processing sequence is resumed, restore this information, and cause the return to the interrupted program.

SVI | STORE VARIANT AND INDICATORS

FORMAT

a.

Op Code	Address Field	Address Field	Variant
46			V

FUNCTION

The SVI instruction is used to store information regarding the current status of the processor when an interrupt condition occurs. The instruction stores the designated information in up to six consecutive locations following its own variant character.

Each bit in the 6-bit variant character (V6V5V4V3V2V1) represents processor control registers or indicators whose contents are to be stored in a single character location. The programmer specifies the amount of information to be stored by selecting the desired entries from Table 6-16 and encoding the resulting bit configuration as two octal digits.

Table 6-16. Information Stored by SVI Instruction

VARIANT CHARACTER $V_6V_5V_4V_3V_2V_1$	INFORMATION STORED
X X X X X 1	The contents of the variant register.
X X X X 1 X	The settings of the arithmetic, comparison, address mode, and item-mark trap mode indicators. This information is stored in seven bit positions of the character location — the six data bit positions and the item-mark bit position. The arithmetic and comparison indicators are cleared when their contents have been stored.
X X X 1 X X	The contents of the auxiliary indicators register (AIR). The contents of the arithmetic, comparison, address mode, and item mark trap mode indicators are stored automatically in this register upon the occurrence of an external interrupt. Upon executing an RNM instruction to return to either standard or internal interrupt mode, the specified indicators are reset automatically using the contents of this register. The contents of this register can be changed by using the RVI instruction (see page 6-94). The auxiliary arithmetic and comparison indicators are cleared when their contents have been stored.

Table 6-16 (cont). Information Stored by SVI Instruction

VARIANT CHARACTER $V_6V_5V_4V_3V_2V_1$	INFORMATION STORED
X 1 X X X X	The settings of the protect and proceed indicators and (if the processor is in the external interrupt mode) the setting of the internal interrupt (II) mode indicator. The protect and proceed indicators are cleared when their contents are stored.
1 X X X X X	The settings of the interrupt source indicators. The stored settings of these indicators can be tested to determine the status of the processor as follows: 1. Whether the processor is in the external interrupt mode, the internal interrupt mode, or the standard processing mode. 2. The source of the interruption if the processor is in the external interrupt mode; any of three sources can be determined — a peripheral control, the console, or the Monitor Call instruction (see page 6-95). 3. Whether an external interrupt (EI) address violation[1] has occurred (if the processor is in the external interrupt mode). 4. Whether an op code violation[1] has occurred (if the processor is in the internal interrupt mode). 5. Whether an internal interrupt (II) address violation[1] has occurred (if the processor is in the internal interrupt mode). The indicators referred to in 3 through 5, above, as well as those which identify the console and the Monitor Call instruction as the interrupt source, are cleared when their contents are stored.
[1] EI address violation, op code violation, and II address violation are associated with the storage protect capability.	

WORD MARKS

A word mark is required in the location following the variant character to terminate the extraction of the SVI instruction. Other word marks (if any) in which information is stored are ignored and unaffected.

ADDRESS REGISTERS AFTER OPERATION

SR	AAR	BAR
NXT	A_p	B_p

NOTES

1. Only the number of characters specified by the variant character are stored. They are stored in the order listed in Table 6-17; the contents of the variant register (if specified) are stored in the location immediately following the SVI instruction, etc., using only those locations actually required to store the requested information.
2. Item-mark and data-bit positions which are not used to store information are cleared to zeros.
3. The format in which information is stored by the SVI instruction is shown in the following table. Indicators which are cleared (i.e., set to zero) when their contents are stored are indicated by an asterisk (*).
4. The current status of the arithmetic, comparison, address mode, and trap mode indicators are not stored in the auxiliary indicators register (AIR) when an internal interrupt occurs. The contents of the AIR should therefore not be stored by an SVI instruction in the internal interrupt mode, for the contents of the AIR would be meaningless at the time of internal interruption.
5. The SVI op code is a "privileged" op code that has special significance in the VLF processor when storage protection is in effect (see Appendix F).
6. The method of coding interrupt service routines is described in Appendix E, "Interrupt Processing."
7. The contents of the variant register are not altered by the execution of this instruction; i.e., the variant character of the SVI instruction is not stored therein.

EXAMPLE

Store the following information in the three successive memory locations which immediately follow the variant character of the instruction:

1. The contents of the variant register;
2. The contents of the auxiliary indicators register (AIR); and
3. The settings of the interrupt source indicators.

The op code of the SVI instruction is tagged STORE, so that the locations of the stored information are STORE+2, STORE+3, and STORE+4.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ____ OF ____

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1			STORE	SVI	4,5	
2						
3						

Table 6-17. Variant Character Storage

VARIANT BIT	STORED CHARACTER LOCATION BITS						
	I/M BIT	B BIT	A BIT	8 BIT	4 BIT	2 BIT	1 BIT
V_1	0 Item-mark	Contents of Variant Register					
V_2	Trap-mode: 1=yes; 0=no.	Address mode: 01=2-character; 00=3-character; 11=4-character.		Overflow: 1=yes; 0=no. *	Zero Balance: 1=yes; 0=no. *	A ≤ B: 1=yes; 0=no. *	A = B; 1=yes; 0=no. *
V_3	Contents of AIR (identical to information in V_2, above)			*	*	*	*
V_4	0	MPO:* 1=yes; 0=no.	DVC:* 1=yes; 0=no.	EXO:* 1=yes; 0=no.	Sector 0 Interrupt Mask 0=off; 1=on.	Sector 1 Interrupt Mask 0=off; 1=on.	Sector 2 Interrupt Mask 1=off; 0=on.
V_5	0	Protect indicator: 1=on; 0=off. *	Instruction timeout allow 1=on; 0=off.*	S mode	Proceed indicator: 1=on; 0=off. *	Relocation indicator: 1=on; 0=off.	In external interrupt mode only: 1=II indicator on; otherwise, 0
V_6**	Processor is in external interrupt mode						
	0	EI Address violation: 1=yes; 0=no. *	Monitor Call: 1=yes; 0=no. *	Console interrupt: 1=yes; 0=no. *	Peripheral interrupt: 1=yes; 0=no.	1	II Mode indicator: 1=on; 0=off.
	Processor is in internal interrupt mode						
	0	II Address violation: 1=yes; 0=no. *	Op code violation: 1=yes; 0=no. *	Instruction: timeout 1=on; 0=off. *	0	0	1

*These indicators are cleared when their contents are stored.
**The character specified by V_6 is not restored by the RVI instruction.

RVI	RESTORE VARIANT AND INDICATORS

FORMAT

a.

Op Code	Address Field	Address Field	Variant
67	A		V

FUNCTION

Up to five consecutive characters (previously stored via an SVI instruction) are loaded into the processor control registers and/or indicators specified by the variant character. Characters are retrieved from left to right, beginning with the character specified by the A address.

The low-order five bits of the variant character specify the registers and/or indicators whose contents are to be restored. The programmer specifies the amount of information to be restored by selecting the desired entries from Table 6-18 and encoding the resulting bit configurations as two octal digits.

Table 6-18. Information Restored by RVI Instruction

VARIANT CHARACTER $V_6V_5V_4V_3V_2V_1$	INFORMATION RESTORED
0 X X X X 1	The contents of the variant register.
0 X X X 1 X	The settings of the arithmetic, comparison, address mode, and item-mark trap mode indicators. This information is stored in the six data bits and the item-mark bit of a character location.
0 X X 1 X X	The contents of the auxiliary indicators register (AIR). Upon returning from external interrupt mode to either internal interrupt or standard mode, the contents of this register are moved automatically to the indicators specified above for V_2.
0 1 X X X X	The settings of the protect and proceed indicators and (if the processor is in the external interrupt mode) the setting of the internal interrupt (II) mode indicator.

WORD MARKS

Word marks neither affect nor are affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

SR	AAR	BAR
NXT	A_p	B_p

NOTES

1. Each entry in the righthand column of Table 6-18 is retrieved from a single character location. Only the number of characters corresponding to the selected table entries are retrieved by the RVI instruction.
2. The RVI op code is a "privileged" op code that has special significance in the VLF processor when storage protection is in effect (see Appendix F).
3. The format in which information is stored by an SVI instruction is shown in Table 6-16. Note that the information contained in the last character location is not restored by the RVI instruction.
4. The method of coding interrupt service routines is described in Appendix E, "Interrupt Processing."
5. The protect and proceed indicators are not turned on automatically by the computer but instead must be turned on by programmed instructions, as follows: (1) a 1-bit is set in the bit position which, when restored by the RVI instruction, indicates the status of the indicator; and (2) an RVI instruction with the V5 bit set to 1 in the variant character is executed, thereby turning on the appropriate indicator.
6. Unless the contents of the variant register are explicitly restored by this instruction, they are not altered by its execution, i.e., the variant character of the RVI instruction is not stored in the variant register.

EXAMPLE

Restore the contents of the variant register and auxiliary indicators register (AIR) that were previouly stored by the SVI instruction example on page 6-90.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ___ _____ PAGE ___ OF ___

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS
1 2 3 4 5 6	7	8	14 15	20 21	62 63 80
1				RVI	STORE+2,Ø5
2					
3					
4					

MC | MONITOR CALL

FORMAT

	Op Code	Address Field	Address Field
a.	44		

FUNCTION

The Monitor Call instruction causes the processor to enter the external interrupt mode (if the processor is not already in that mode). The following activities are automatically performed:

1. The EI interrupt source indicators are set to show that the Monitor Call instruction is the source of interruption, and the processor enters the external interrupt mode;
2. The settings of the arithmetic, comparison, address mode, and item-mark trap mode indicators are stored in the auxiliary indicators register (AIR);
3. The arithmetic indicators are cleared;
4. The contents of the sequence register (SR) and the external interrupt register (EIR) are interchanged, and the program branches to the instruction whose op code address was previously stored in EIR;
5. The processor switches to the 3-character, non-trap mode.

WORD MARKS

Word marks are not affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

SR	EIR	AAR	BAR
JI (contents of EIR)	NXT	A_p	B_p

NOTES

1. If this instruction is issued in the external interrupt mode, the results are unspecified.
2. The interrupt source indicators can be stored via an SVI instruction. Their stored contents can then be interrogated by programmed instruction to determine the interrupt source.

EXAMPLE

Interrupt the central processor and branch to MONTOR, the location of the monitor program. The address tagged MONTOR was previously stored in EIR.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ____ OF ____

	CARD NUMBER	TYPER	MDR	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1					SCR	MONTOR,66	
2					≀		
3					≀		
4					≀		
5					MC		
6							

RNM	RESUME NORMAL MODE

FORMAT

	Op Code	Address Field	Address Field
a.	41	A	B
b.	41	A	
c.	41		

FUNCTION

Format a: The RNM instruction causes an exit from the program being executed in the interrupt mode (external or internal) to the program which was interrupted. The activities performed depend on the type of interrupt mode in which the instruction is issued.

When the RNM instruction is issued in the external interrupt mode:

1. The EI mode indicators are turned off;
2. The arithmetic, comparison, address mode, and item-mark trap mode indicators are restored to the status specified by the auxiliary indicators register (AIR);
3. The A and B addresses of the RNM instruction are stored in the A- and B- address registers (AAR and BAR), respectively; and
4. The contents of the sequence register (SR) and the external interrupt register (EIR) are interchanged, and the program branches to the instruction whose op code address was initially stored in EIR when the external interrupt occurred.

When the RNM instruction is issued in the internal interrupt mode:

1. The II mode indicator is turned off;
2. The A and B addresses of the RNM instruction are stored in AAR and BAR, respectively; and
3. The contents of SR and the internal interrupt register (IIR) are interchanged, and the program branches to the instruction whose op code address was initially stored in IIR when the internal interrupt occurred.

Format b: This format operates like format a. except that the B address of the RNM instruction is not stored in BAR. The previous contents of BAR are not changed.

Format c: This format operates like format a. except that no instruction addresses are stored. The previous contents of AAR and BAR are not affected by this format.

WORD MARKS

Formats a, b, and c:

Word marks are not affected by this instruction.

ADDRESS REGISTERS AFTER OPERATION

	SR	EIR	IIR	AAR	BAR	
Format a:	NXT	address of op code following RNM instruction	n/a	A	B	RNM ISSUED IN EXTERNAL INTERRUPT MODE
	NXT	n/a	address of op code following RNM instruction	A	B	RNM ISSUED IN INTERNAL INTERRUPT MODE
Format b:	NXT	address of op code following RNM instruction	n/a	A	B_p	RNM ISSUED IN EXTERNAL INTERRUPT MODE
	NXT	n/a	address of op code following RNM instruction	A	B_p	RNM ISSUED IN INTERNAL INTERRUPT MODE
Format c:	NXT	address of op code following RNM instruction	n/a	A_p	B_p	RNM ISSUED IN EXTERNAL INTERRUPT MODE
	NXT	n/a	address of op code following RNM instruction	A_p	B_p	RNM ISSUED IN INTERNAL INTERRUPT MODE

NOTES

1. The method of coding interrupt service routines is described in Appendix E, "Interrupt Processing."
2. The RNM op code is a "privileged" op code that has special significance in the VLF processor when storage protection is in effect (see Appendix F).

EXAMPLE

The following simplified coding shows a convenient method of restoring the starting address of the external interrupt routine (EXT2) in EIR when the normal program sequence is resumed.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE ____ PAGE ____ OF ____

CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
			RESUME	RNM		
			EXT2	SV1	45	
					INTERRUPT ROUTINE	
				B	RESUME	

EDITING INSTRUCTIONS

MCE	MOVE CHARACTER AND EDIT

FORMAT

	Op Code	Address Field	Address Field
a.	74	A	B
b.	74	A	
c.	74		

FUNCTION

Format a: The MCE instruction is used to insert identifying symbols and punctuation and to suppress unwanted zeros in a data field. The A field of an MCE instruction contains the information to be edited. The B field contains an edit control word which provides a framework for the edit operation. When an MCE instruction is executed, the data in the A field is moved to the B field where it is punctuated and formatted according to the edit control word already stored in that field.

NOTE: An LCA instruction can be used to load the control word into the field where the edited information will eventually go. For instance, if the edited information is to be printed, the control word should be loaded into the print image area and the address of this area should be used as the B address of the MCE instruction.

Editing is performed according to the following rules:

RULE 1. Any character in the Series 200 character set can be used in the edit control word. Those characters having special meanings are listed in Table 6-19. Any other character, if included in the edit control word, remains in the edited result in the position where written.

RULE 2. A word mark in the high-order position of the B field controls the edit operation.

RULE 3. The number of replaceable characters in the edit control word must be at least as large as the number of characters in the A field.

RULE 4. Data is transferred from the A field character by character, from right to left. If a zero suppression symbol is not sensed in the edit control word, the edit operation terminates when the B-field word mark is sensed. A zero suppression symbol causes the edited result field to be scanned from left to right. During this scan, high-order zeros and commas are automatically replaced by blanks (unless an asterisk appears immediately to the left of the zero suppression symbol - see rule 5). Zero suppression is terminated by any of the following:

a. a decimal digit from 1 through 9,

b. a decimal point, or

c. the location that initially contained the zero suppression symbol.

RULE 5. An asterisk immediately to the left of the zero suppression symbol in the control word causes high-order zeros and commas to be replaced by asterisks instead of blanks in a zero suppression operation. High-order blanks are also replaced by asterisks.

RULE 6. A dollar sign immediately to the left of the zero suppression symbol in the control word is replaced with an A-field character and causes the edited result to be rescanned following the zero suppression operation. During this scan, the dollar sign is "floated" to the left of the high-order significant digit in the edited result.

Table 6-19. Edit Control Word Special Characters

CONTROL CHARACTER	FUNCTION
b (blank)	Blanks are replaced with A-field characters such that the rightmost character in the A field replaces the rightmost blank in the edit control word and all higher-order A-field characters replace successively higher-order blanks.
0 (zero)	This symbol specifies zero suppression. Its location in the control word is interpreted as the rightmost limit of zero suppression. It is replaced with an A-field character.
. (decimal point)	The decimal point remains in the edited field in the position where written.
, (comma)	Commas remain in the edited field where written unless zero suppression is specified (see rule 4). Commas in control word positions to the left of the high-order character transferred from the A field are replaced by blanks.
C_R, CR (credit) $\bar{0}$ (minus) NOTE: $\bar{0}$ is printed as a minus symbol.	The credit or minus symbol is undisturbed if the sign in the units position of the A field is negative. If the sign is positive, the credit (or minus) symbol is blanked out. A credit (or minus) symbol transferred from the A field is not subject to sign control.
37_8	An octal 37 is replaced by a blank in the edited field.

Table 6-19 (cont). Edit Control Word Special Characters

CONTROL CHARACTER	FUNCTION
* (asterisk)	The asterisk remains in the edited field in the position where written unless it appears immediately to the left of the zero suppression symbol (see rule 5).
$ (dollar sign)	The dollar sign remains in the edited field in the position where written unless it appears immediately to the left of the zero suppression symbol (see rule 6).

Format b: The data contents of the A field are edited and stored in the field specified by the contents of the B-address register according to the rules outlined previously.

Format c: The data field specified by (AAR) are edited and stored in the field specified by the contents of BAR according to the rules outlined previously.

WORD MARKS

Formats a, b, and c:

Both the A field and the B field must have defining word marks. The A-field word mark terminates the transfer of data from the A-field. The B-field word mark terminates the edit operation if no zero suppression symbol is sensed in the edit control word or if automatic dollar sign insertion is specified in conjunction with zero suppression. The B-field word mark is erased after terminating the edit.

If zero suppression is specified, a word mark is automatically set in the location containing the zero suppression symbol. When this word mark is sensed during the reverse scan associated with the zero suppression operation, it is erased and, if automatic dollar sign insertion is not called for, the edit operation terminates.

ADDRESS REGISTERS AFTER OPERATION

NOTES Unspecified.

1. The zone bits in the units position of the A field are cleared to zero when moved to the B field. Therefore, the value of the character in the units position in the A field may change when moved to the B field. For example, an F in the units position of the A field will appear as a 6 in the result field.
2. Floating dollar sign insertion and automatic insertion can not be performed in the same edit operation.
3. The contents of the variant register are unspecified following the execution of this instruction. Therefore, an instruction requiring a variant character cannot be chained following an MCE instruction.

EXAMPLES[1]

Data Field (A Field)		0 00009$\bar{9}$
Control Word (B Field)	b	bb,bb0.bb&&$\bar{0}$
Result of Edit		.99

Example 1.

Data Field (A Field)		2 5454986
Control Word (B Field)	b	bb&bb&bbb
Result of Edit		254 54 986

Example 2.

Data Field (A Field		0 00450$\dot{+}$
Control Word (B Field)	$	b,bb0.bb&CR*
Result of Edit	$	4.50 *

Example 3.

Data Field (A Field)		0 0897445
Control Word (B Field)	b	bbb,b$0.bb
Result of Edit		$8,974.45

Example 4.

Data Field (A Field		0 010450
Control Word (B Field)	b	b,b*0.bb
Result of Edit		***104.50

Example 5.

INPUT/OUTPUT CONTROL OPERATIONS

Effective control over data transfers between the VLF processor and peripheral units and over the peripheral units themselves is maintained by the use of two basic instructions: Peripheral Data Transfer (PDT), and Peripheral Control and Branch (PCB). The PDT instruction is used to initiate data transfer operations and certain other related operations, such as backspace magnetic tape and advance the printer form.

[1] The character (37_8) is shown as an ampersand (&) in these examples.

The PCB instruction can perform four distinct functions: (1) it initiates strictly mechanical (non-data transfer) operations such as magnetic tape rewinds and card rejections; (2) it causes a program branch to be performed contingent upon the settings of peripheral condition indicators; (3) it changes the operational mode of a peripheral control; and (4) it allows a peripheral control to interrupt (or directs the control not to interrupt) the VLF processor when data transfer is completed.

The remainder of this section (in conjunction with Appendix C) is a summary of the PDT and PCB instructions. In all applicable cases, the coding summary for a device is followed by a reference to the specific Honeywell manual or information bulletin where additional information can be found.

PDT | PERIPHERAL DATA TRANSFER

FORMAT

	Op Code	Address Field	I/O	Control Characters
a.	66	A	C1	C2 C3....Cn

FUNCTION

Format a: The PDT instruction causes data to be transferred between a peripheral device and the main memory area whose leftmost location is designated by the A address. Data transfer is terminated according to the data medium employed. Input/output control characters specify the data path through which the transfer is to be accomplished, as indicated in Table 6-20.

Table 6-20. Description of PDT I/O Control Character C2 (Peripheral Control Designation)

CONTROL CHARACTER	DESCRIPTION
C2	PERIPHERAL CONTROL DESIGNATION: This 6-bit character specifies the logical address of the peripheral control to be used in the data transfer. C2 X X X X X X Peripheral Control Address Bits Sector Bits Input/Output Bit

Table 6-20 (cont). Description of PDT I/O Control Character C2 (Peripheral Control Designation)

CONTROL CHARACTER	DESCRIPTION
	Input/Output Bit: This bit specifies the direction of data transfer when a peripheral control capable of both reading and writing is involved in the transfer. When such a bidirectional control is used, 0 = transfer data from memory to the peripheral control (output), 1 = transfer data to memory from the peripheral control (input). Sector Bits: These bits specify the sector in which the peripheral control is connected. They are specified as follows: Sector 1 00 Sector 2 10 Sector 3 11 Peripheral Control Address Bits: These three bits, in conjunction with the preceding three bits, identify the address of the peripheral control involved in the operation.

Additional Parameters (C3 through Cn)

The specific use of these control characters is dependent upon the type of peripheral device addressed. A summary of coding for these characters may be found in Appendix C.

PUNCTUATION MARKS

The execution of this instruction neither affects nor is affected by word marks or item marks. However, record marks may terminate the data transfer, depending upon the device used and the operation performed.

ADDRESS REGISTERS AFTER OPERATION

SR	AAR	BAR
NXT	A	B_p

NOTES

1. If either the read/write channel or the peripheral control (specified by C1 and C2, respectively) is found "busy" during the extraction of a PDT instruction, the instruction is reextracted: the contents of SR are set back to the address of the PDT op code, and the extraction process begins again.
2. The PDT op code is a "privileged" op code that has special significance in the VLF processor when storage protection is in effect (see Appendix F).

3. Unspecified processor activity can occur when an attempt is made to execute a PDT instruction having a read/write channel assignment (C1) of zero. It is therefore imperative that every PDT instruction contain some legal RWC assignment.

4. Control character C1 of a PDT instruction is stored in the variant register.

EXAMPLE

Read a card into the 80-character image area tagged CREAD. Use RWC2 and assume that the card reader control is assigned to the logical address of octal 41. Note that the data transfer rate in a card reading operation is less than 83,300 characters per second.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

	CARD NUMBER	TYPE	MAR	LOCATION	OPERATION CODE	OPERANDS
	1 2 3 4 5	6	7 8 ... 14	15 ... 20	21 ... 62	63 ... 80
1				PDT	CREAD,12,41	
2						

PCB	PERIPHERAL CONTROL AND BRANCH

FORMAT

	Op Code	Address Field	I/O Control Characters
a.	64	A	C1 C2 C3....Cn
b.	64	A	C1

TYPES OF TEST AND CONTROL OPERATIONS

The Peripheral Control and Branch instruction can initiate four types of operations: (1) strictly mechanical peripheral device operations; (2) test and branch operations; (3) mode change operations; and (4) peripheral interrupt operations.

1. A mechanical operation is a non-data transfer operation such as rewind magnetic tape or seek a Disk Pack Drive cylinder.

2. A test and branch operation tests the status of a peripheral control and/or a read/write channel(s). If the condition being tested (e.g., peripheral control busy, error in last card punched) is present, a program branch is performed.

3. A mode change operation conditions the addressed peripheral control to operate in a specific mode. For instance, the card reader control can be conditioned to reject illegally punched cards, to generate a busy signal if illegally punched cards are read, or both, depending upon the control characters of the PCB instruction.

4. A peripheral interrupt operation directs a peripheral control to change the setting of an interrupt function or an allow interrupt function (see Appendix E).

Control character C1 designates a read/write channel or combination of channels whose busy status is to be tested. If an RWC busy test is not desired, C1 must contain zeros. C2 designates the logical address of the peripheral control to be tested or actuated. The coding of this character is the same as its coding for a PDT instruction (see Table 6-20).

Control characters C3 through Cn designate the control and test operations. Any number of control characters may follow C2, each one designating a different operation. If control characters within a single instruction designate conflicting operations (e.g., punch Hollerith code and punch direct transcription mode), the control character to the left is cancelled by a conflicting control character to the right within the same instruction. If multiple test operations are specified within a single instruction, a branch will occur if any of the conditions tested is present. The specific use of characters C3 through Cn is dependent upon the type of peripheral device addressed. Tables C-8 through C-10 summarize the coding of these characters.

FUNCTION

Format a: The read/write channel or channel combination specified by C1 is tested for busy status. If it is busy, a branch is made to the instruction at A. If the RWC is not busy (or if C1 is 00_8), the operation(s) specified by characters C3 through Cn is performed on the peripheral control specified by C2. This peripheral control must be connected to the input/output sector implied by the value of C1.

Format b: The read/write channel or channel combination specified by C1 is tested for busy status. If it is busy, a branch is made to A. If the RWC is not busy, the instruction following the PCB is executed.

PUNCTUATION MARKS

The execution of this instruction neither affects nor is affected by word marks or record marks.

ADDRESS REGISTERS AFTER OPERATION

SR	AAR	BAR	
NXT	A	B_p	NO BRANCH
JI (A)	A	NXT	BRANCH

NOTES

1. The PCB op code is a "privileged" op code that has special significance in the VLF processor when storage protection is in effect (see Appendix F).
2. Control character C1 of a PCB instruction is stored in the variant register.

EXAMPLE

In the following example, assume that the logical address of the card reader control is octal 41.

Set the card reader control to read Hollerith code (C3 = 27) and to reject automatically all cards with hole-count errors (C4 = 21). If the device is inoperable, branch to the location tagged STOP. (Note that since an RWC is not to be tested, C1 must contain zeros.)

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 ... 14	15 ... 20	21 ... 62	63 ... 80
1					PCB	STOP,ØØ,41,27,21	
2							
3							

APPENDIX A
SCIENTIFIC INSTRUCTIONS

The 19 scientific instructions include 18 instructions that manipulate data in floating-point form, and one that converts data between fixed-point decimal and floating-point binary form. In a system that includes the Scientific Unit (Feature 8201-B), these 19 instructions are executed as machine instructions. In a system not equipped with Feature 8201-B, they are interpreted as pseudo instructions that call in library routines to perform the desired operations.

Floating-Point Numbers

A number can be represented in floating-point form by a mantissa (signed proper fraction) and an exponent (integral power of the radix), allowing the computer to keep track of the radix points and greatly reducing the problem of overflow. A floating-point number can be converted to fixed-point form by obtaining the product of the mantissa and the radix raised to the power of the exponent.

In Section II, the structure of a Model 8200 floating-point word is shown as a 1-bit sign, a 7-bit exponent, and a 40-bit mantissa. In a floating-point decimal word, the mantissa is interpreted as 10 decimal digits; in a floating-point binary word, it is interpreted as 40 binary (or 10 hexadecimal) digits. A binary one in the sign position indicates that the number is positive, a binary zero that it is negative. The seven bits of the exponent can represent 128 different exponent values. In order to store floating-point words having negative as well as positive exponents, the actual exponent is defined as a number 64 less than the effective (or working) exponent stored in these seven bits. Thus, the range of the actual exponent in a floating-point word is from -64 to +63 as shown in the following table. In a floating-point word, the actual exponent is interpreted as a power of 10; in a floating-point binary word, it is interpreted as a power of 16.

Effective Exponent		Actual Exponent
(binary)	(decimal)	(decimal)
0000000	0	-64
0111111	63	-1
1000000	64	0
1111111	127	+63

In floating-point form, there is no single number that uniquely represents a given fixed-point number. For example, the fixed-point decimal number .000563 can be represented in floating-point decimal form as:

$.000563 \times 10^{0}$;

or $.0563 \times 10^{-2}$;

or $.563 \times 10^{-3}$; etc.

The normalized (or normal) form for expressing floating-point numbers is with the implied indicator point immediately to the left of the first significant digit. Therefore, the third example above shows the number .000563 in normalized floating-point form. The word processor would store this normalized number as follows:

1 0111101 0101 0110 0011 0000 0000 0000 0000 0000 0000 0000

which represents a positive sign, an effective exponent of 61 (actual exponent -3), and a mantissa of 5630000000. (Note that in floating-point binary, the same configuration represents the value $+.01010110001100\text{-------}00 \times 16^{-3}$.)

Since a normalized decimal mantissa may range from .1000000000 up to .9999999999, the range of a normalized floating-point decimal word is from $.1 \times 10^{-64}$ (or 10^{-65}) up to $.9999999999 \times 10^{63}$ (or virtually 10^{63}). In the same manner, the range of a normalized floating-point binary word is from $.0001 \times 16^{-64}$ (or 16^{-65}) up to $.1111\text{----}11 \times 16^{63}$ (or virtually 16^{63}).

The number zero can be represented by any floating-point number in which the mantissa is zero. A normalized zero, however, is defined in the word processor by the configuration 1000 0000 .... 0000, which represents a positive sign, a working exponent of zero (or actual exponent of -64), and a mantissa of zero. Any instruction which produces normalized results will deliver a high-order result of zero in this form.

The operands of floating-point instructions do not have to be normalized (except for floating-point divisors and comparison operands). The results are normalized, with the exception of:

1. Quotients of division operations involving non-normalized dividends;
2. Sums and differences of unnormalized addition and subtraction; and
3. The low-order portions of double-precision results.

If non-normalized operands are used in an operation which produces a normalized result, a significant digit may be lost in the result for every high-order zero in the operands.

A number that can be stored in one machine word is called a single-precision number; one that requires two machine words is known as a double-precision number. A double-precision floating-point number is defined as two single-precision numbers in which the signs are identical and the exponent of the low-order portion is ten less than the exponent of the high-order portion. A double-precision number is considered normalized if the high-order portion is in normalized form.

Certain floating-point instructions in the word processor always produce double-precision results. If the high-order portion of such a result is zero, it is produced in normalized form and the low-order portion is also a normalized zero. If, however, the high-order portion is not zero and the low-order portion is zero, the latter will not be normalized; i.e., the low-order portion will have an exponent ten less than that of the high-order portion. This non-normalized zero will, however, be treated as normalized if it is used in any instruction which produces a normalized result. Care must be exercised if this non-normalized zero is used in a comparison instruction in which the other operand is a normalized zero.

Floating-point data words are identified in Mod 8 assembler language by the constant codes FLDEC, floating-point decimal number; FLBIN, floating-point binary number; and EBC, extended binary number. A floating-point number may be specified with an indicator point, an explicit exponent, or both. An explicit exponent is expressed as an "E" and a signed or unsigned exponent immediately following the number. A floating-point decimal number may consist of up to ten signed or unsigned decimal digits and may have any value within the acceptable range. A floating-point binary number may consist of up to 13 signed or unsigned decimal digits and may have any value within the acceptable range. The assembler converts these numbers to normalized floating-point decimal and binary words, respectively. An extended binary number may consist of up to 25 signed or unsigned decimal digits and may have any value within the acceptable range for floating-point binary words. The Mod 8 assembler converts this number to normalized double-precision floating-point binary form, retaining 80 bits of the mantissa. The high-order 40 bits are stored with proper exponent and sign as one machine word; the low-order 40 bits are stored, with the same sign and an exponent 10 less than that of the high-order word, as the following word.

Instruction Configurations

The operation codes for the 19 scientific instructions are uniquely designated by command ~~code bits~~ 2, 3, and 7 through 12. Bit 1 is the bisequence bit, and bits 4 through 6 are the A, B, and C memory designator bits, respectively. All six types of addressing (in both normal and extended mode) can be used with the scientific instructions. In most cases, however, only the four types that address main memory are meaningful, due to the nature of the floating-point word. The scientific instructions comprise the following groups: floating addition, floating subtraction, floating multiplication, floating division, floating comparison, normalization, conversion, and Multiple Unload.

Inactive Addresses

Except where otherwise specifically stated below, any address or combination of addresses may be inactive in a scientific instruction. In general, if the C address is active, the contents of

the accumulator are delivered to the location specified by C and the system hunts for the next sequencing counter in demand. Exceptions are the Multiple Unload and the comparison instructions. If the C address is inactive, hunting is inhibited to permit retrieval of the contents of the accumulator and, if desired, of the low-order product register.

The general rule for an inactive A or B address or both is as follows: the operand represented by the inactive address is replaced by the contents of the accumulator (including the sign and exponent). For example, if A is inactive, then the accumulator is treated as the A operand. Note that an addition instruction with inactive A and B addresses may be used to multiply the contents of the accumulator by two, while a multiply with inactive A and B addresses may be used to square the contents of the accumulator. Here the exceptions are the Multiple Unload and Fixed-to-Floating Normalize instructions.

Exponential Overflow and Underflow

Although the use of floating-point numbers greatly reduces the problem of overflow, it does not entirely resolve this problem. Right shifts occur automatically to counter any mantissa overflow in a result, but exponential overflow occurs if the exponent of a result exceeds +63, i.e., if the working exponent is greater than +127. Exponential underflow, on the other hand, occurs if the result exponent drops below -64, i.e., if the working exponent is less than zero. Either condition results in an unprogrammed transfer of control. If exponential overflow occurs, the instruction is stored in U or U + 1, depending on which sequencing counter selected it, and the next instruction is taken from U + 14 or U + 15 (see Table 5-3, page 5-16). The exponent of the result is reduced by 128, bringing it within the acceptable range. In other words, if one is added to a working exponent of 127 (actually +63), the result will have a working exponent of zero (actually -64). If exponential underflow occurs, the instruction is stored in U or U + 1, the next instruction is taken from U + 12 or U + 13, and the result exponent is increased by 128.

Exponential overflow or underflow in a result which is deliverable to the location specified by C normally causes an immediate unprogrammed transfer, whether the C address is active or inactive. An exception occurs in the case of an instruction which uses the previous contents of the accumulator as an operand and which does not deliver a result to main memory (inactive C address). If exponential overflow or underflow occurs in such a case, an overflow or underflow indicator is set. The unprogrammed transfer is delayed until the first occurrence of one of the following:

1. A scientific instruction does not use the previous contents of the accumulator as an operand;
2. An instruction delivers a result to main memory; or
3. The actual exponent of the result becomes greater than +127 or less than -128.

Upon the occurrence of an overflow unprogrammed transfer, the programmer can determine whether the unprogrammed transfer was caused by a "normal" overflow (i.e., exponent greater than +63) or by a "double" overflow (i.e., exponent greater than +127) by examining the high-order bit of the result exponent. Figure A-1 shows the ranges of exponents for both valid floating-point numbers and stored overflow and underflow conditions. Note that following an unprogrammed transfer for normal overflow the high-order bit of the exponent is a zero, whereas following an unprogrammed transfer for double overflow this bit is a one. The high-order bit of the exponent can be used in the same manner following an underflow unprogrammed transfer.

The floating-point division instructions do not conform to the above discussion of stored overflow and underflow. In the event of overflow or underflow in the quotient of such an instruction, the unprogrammed transfer immediately follows the completion of the instruction. Neither condition is ever stored.

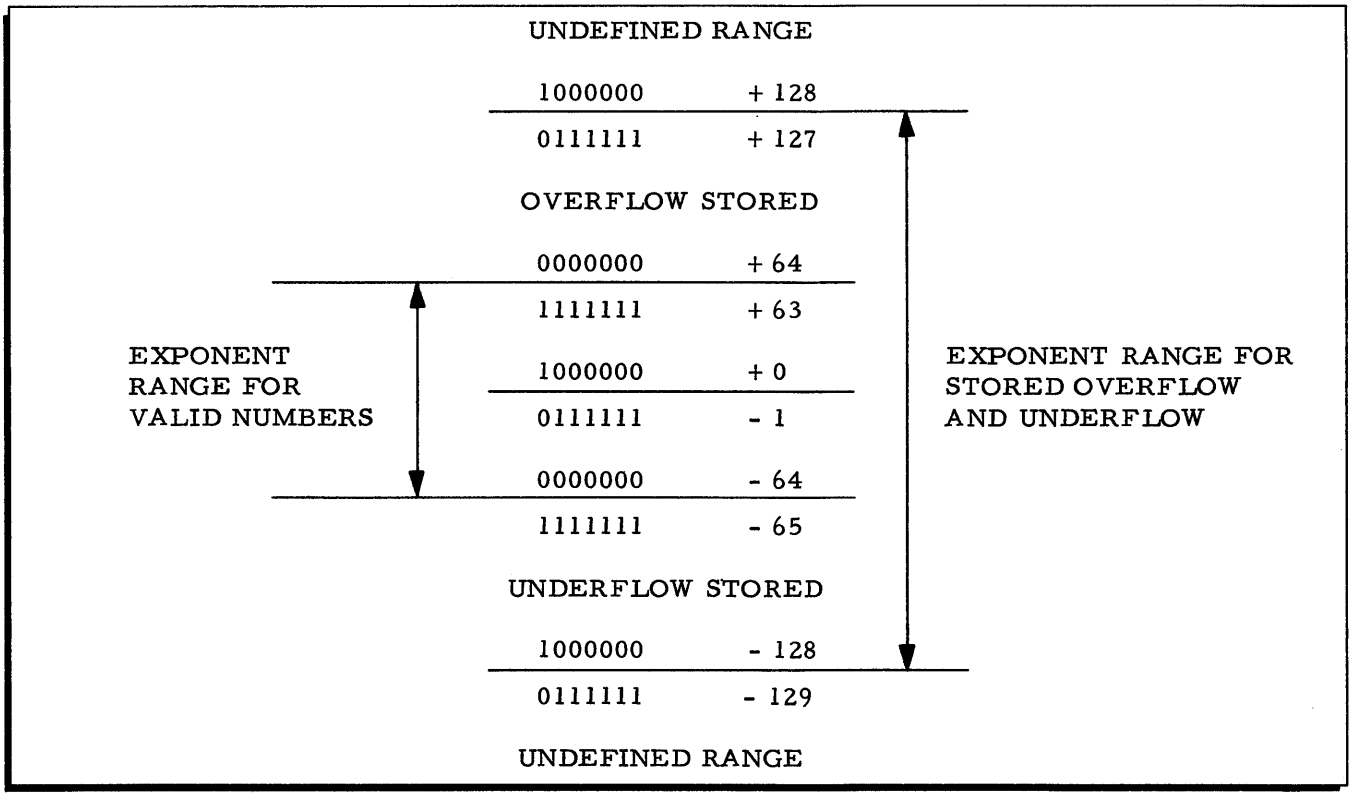


Figure A-1. Exponent Ranges in the Floating-Point Option

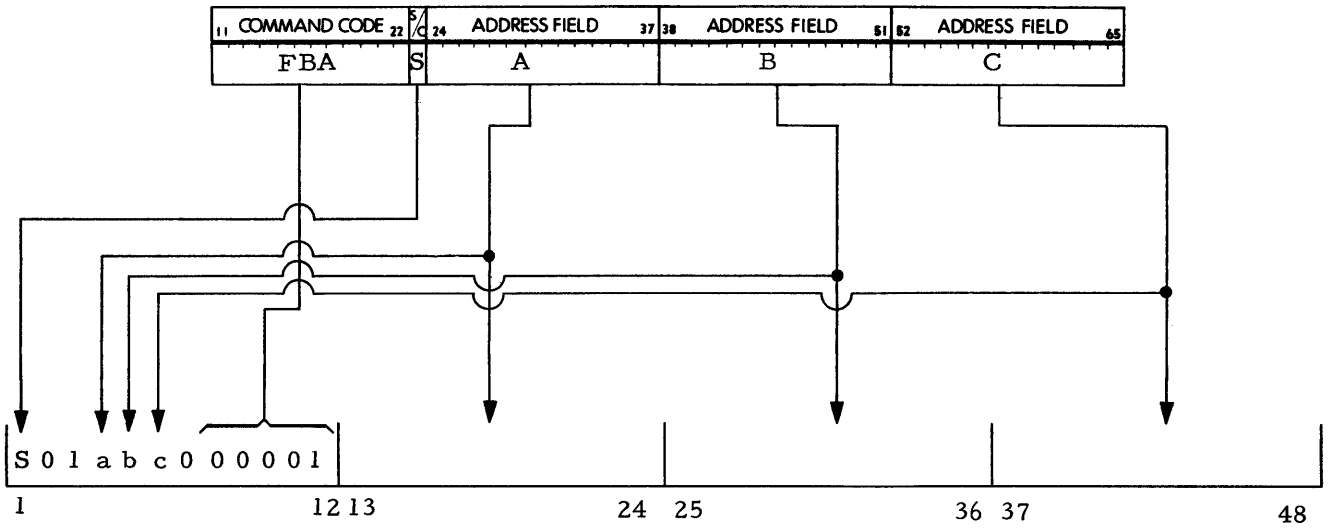
Exponential overflow[1] or underflow in the contents of the low-order product register does not cause an unprogrammed transfer but merely sets an indicator which is reset by the start of

[1] Since the result in the low-order product register usually has a smaller exponent than the result in the accumulator, exponential overflow in the contents of the low-order product register is normally impossible. An exception is the case of a floating-point division instruction which uses the previous contents of the accumulator as an operand, in which case it is possible to have exponential overflow in the remainder.

any instruction except Multiple Unload. Note that whenever a Multiple Unload instruction follows an instruction which used the previous contents of the accumulator as an operand and which did not deliver a result to main memory, an exponential underflow unprogrammed transfer may result from exponential underflow in either the accumulator or the low-order product register.

Assembly

All of the instructions in the scientific grouping are assembled as indicated in the following diagram (using the Floating Binary Add instruction):



Normalized Floating-Point Addition and Subtraction

In the execution of the following six floating-point addition and subtraction instructions, the following stages occur:

1. Prenormalization. During this stage, all zero operands are normalized and the nonzero operand with the larger exponent is normalized, if necessary. If the exponents of the operands become equal during this process, the prenormalization halts.

2. Right Justification. The mantissa of the operand with the smaller exponent is shifted right until its exponent is equal to that of the other operand; this exponent is retained as the tentative exponent of the result. In single-precision instructions, the bits that are shifted off are discarded immediately, but in extended-precision instructions they are discarded only after they have been shifted more than 40 bit positions.

3. Mantissa Arithmetic. Depending upon the operation code and upon whether the operand sign bits are like or unlike, the arithmetic circuits are set to perform an effective addition or subtraction of the mantissas. In the extended-precision instructions, the operation involves an 80-bit mantissa; a 40-bit mantissa is involved in the other floating add and subtract instructions. If mantissa overflow occurs, the mantissa of the result is shifted right four positions and the tentative result exponent is increased by one.

4. Result Normalization. The result is renormalized (if necessary) upon completion of the mantissa arithmetic. Exponential overflow and underflow

are checked after the result exponent has been decreased by one for each 4-bit left shift. If a zero result is obtained from the execution of an extended-precision instruction, normalized zeros are provided for both the high-order and low-order results.

5. Result Transmission. The high-order result is retained in the accumulator and the low-order result, if any, in the low-order product register. If C is active, the high-order result is also transmitted to the location specified by C. Arithmetic and control checks are completed at this time whether C is active or inactive.

FBA	FLOATING BINARY ADD

FORMAT

1 12	13 24	25 36	37 48
Command Code	Address Field	Address Field	Address Field
B01 abc 000 001 $(1001)_8$	A	B	C

FUNCTION

The Floating Binary Add instruction performs an algebraic binary addition of the contents of the location specified by the A-address field and the contents of the location specified by the B-address field, producing as a result a normalized, single-precision, floating-point binary number in C; the system hunts for the next sequencing counter in demands. If C is inactive, the result is not delivered to main memory (it is retained in the accumulator) and hunting is inhibited. Provision is made for automatic handling of exponential overflow and underflow in the following manner: If exponential underflow occurs, take the next instruction from (UTR) + 12 or (UTR) + 13. If exponential overflow occurs, store the current instruction in (UTR) or (UTR) + 1.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Binary Add instruction may use inactive addressing in the manner described in Table A-1, on page A-24.

FDA | FLOATING DECIMAL ADD

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B01 abc 010 001 $(1021)_8$	A	B	C

FUNCTION

The Floating Decimal Add instruction performs an algebraic decimal addition of the contents of the location specified by the A-address field and the contents of the location specified by the B-address field, producing as a result a normalized, single-precision, floating-point number in C; the system hunts for the next sequencing counter in demand. If C is inactive, the result is not delivered to main memory (it is retained in the accumulator) and hunting is inhibited. Provision is made for automatic handling of exponential overflow and underflow in the following manner: If exponential underflow occurs, take the next instruction from (UTR) + 12 or (UTR) + 13. If exponential overflow occurs, take the next instruction from (UTR) + 14 or (UTR) + 15. If either occurs, store the current instruction in (UTR) or (UTR) + 1.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Decimal Add instruction may use inactive addressing in the manner described in Table A-1, on page A-24.

FBS | FLOATING BINARY SUBTRACT

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B01 abc 001 001 $(1011)_8$	A	B	C

FUNCTION

The Floating Binary Subtract instruction performs an algebraic binary subtraction of the contents specified by the B-address field from the contents of the A-address field, producing as a result a normalized, single-precision, floating-point binary number in C; the system hunts for the next sequencing counter in demand. If C is inactive, the result is not delivered to main memory (it is retained in the accumulator) and hunting is inhibited. Provision is made for automatic handling of exponential overflow and underflow in the following manner: If exponential underflow occurs, take the next instruction from (UTR) + 12 or (UTR) + 13. If exponential overflow occurs, take the next instruction from (UTR) + 14 or (UTR) + 15. If either occurs, store the current instruction in (UTR) or (UTR) + 1.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:

	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Binary Subtract instruction may use inactive addressing in the manner described in Table A-1, on page A-24.

FDS | FLOATING DECIMAL SUBTRACT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B01 abc 011 001 $(1031)_8$	A	B	C

FUNCTION

The Floating Decimal Subtract instruction performs an algebraic decimal subtraction of the contents specified by the B-address field from the contents of the location specified by the A-address field, producing as a result a normalized, single-precision, floating-point decimal number in C; the system hunts for the next sequencing counter in demand. If C is inactive, the result is not delivered to main memory (it is retained in the accumulator) and hunting is inhibited. Provision is made for automatic handling of exponential overflow and underflow in the following manner: If exponential underflow occurs, take the next instruction from (UTR) + 12 or (UTR) + 13. If exponential overflow occurs, take the next instruction from (UTR) + 14 or (UTR) + 15. If either occurs, store the current instruction in (UTR) or (UTR) + 1.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Decimal Subtract instruction may use inactive addressing in the manner described in Table A-1, on page A-24.

FBAE | FLOATING BINARY ADD, EXTENDED PRECISION

FORMAT

1 12 13 24 25 36 37 48

Command Code	Address Field	Address Field	Address Field
B00 abc 000 001 $(0001)_8$	A	B	C

FUNCTION

The Floating Binary Add, Extended Precision instruction performs an algebraic binary addition of the contents of the location specified by the A-address field and the contents of the location specified by the B-address field, producing as a result a normalized, double-precision, floating-point binary number in the accumulator and low-order product register. If the C-address field is active, the high-order result is delivered to the location specified and the system hunts for the next sequencing counter in demand. If C is inactive, no result is delivered to main memory (the high-order and low-order parts are retained in the accumulator and low-order product register, respectively) and hunting is inhibited. Provision is made for the automatic handling of exponential overflow and underflow on the result in the accumulator in the following manner: If exponential underflow occurs, take the next instruction from (UTR) + 12 or (UTR) + 13. If exponential underflow occurs, take the next instruction from (UTR) + 14 or (UTR) + 15. If either occurs, store the current instructions in (UTR) or (UTR) + 1. If exponential underflow occurs on the result in the low-order product register, the low-order product underflow indicator is set; the nature of the double-precision result precludes the occurrence of the low-order overflow.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Binary Add, Extended Precision instruction may use inactive addressing in the manner described in Table A-2, on page A-25.

FBSE	FLOATING BINARY SUBTRACT, EXTENDED PRECISION

FORMAT

1 12	13 24	25 36	37 48
Command Code	Address Field	Address Field	Address Field
B00 abc 001 001 $(0011)_8$	A	B	C

FUNCTION

The Floating Binary Subtract, Extended Precision instruction performs an algebraic subtraction of the contents of the location specified by the B-address field from the contents of the location specified by the A-address field, producing as a result a normalized, double-precision, floating-point binary number in the accumulator and the low-order product register. If the C-address field is active, the high-order result is delivered to the location specified and the system hunts for the next sequencing counter in demand. If C is inactive, no result is delivered to main memory (the high-order and low-order parts are retained in the accumulator and the low-order product register, respectively) and hunting is inhibited. Provision is made for automatic handling of exponential overflow and underflow on the result in the accumulator in the following manner: If exponential underflow occurs, take the next instruction from (UTR) + 12 or (UTR) + 13. If exponential overflow occurs, take the next instruction from (UTR) + 14 or (UTR) + 15. If either occurs, store the current instructions in (UTR) or (UTR) + 1. If exponential underflow occurs on the result in the low-order product register, the low-order underflow indicator is set; the nature of the double-precision result precludes the occurrence of low-order overflow.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Binary Subtract, Extended Precision instruction may use inactive addressing in the manner described in Table A-2, on page A-25.

Unnormalized Floating-Point Addition and Subtraction

The operation of the following four floating-point addition and subtraction instructions is identical to that of the normalized instructions described previously, except that stages 1 (prenormalization) and 4 (result normalization) are omitted and exponential overflow and underflow are checked at the completion of stage 3 (mantissa arithmetic).

Unnormalized zero results may be either positive or negative. If the operand of larger exponent is zero, but the operational result is not zero, the result exponent is that of the zero operand while the sign is normally that of the nonzero operand. An exception occurs, however, if the nonzero operand in a subtraction operation appears in the location specified by the B address. In this case, the sign of the result is the opposite of the sign of the nonzero operand.

FBAU	FLOATING BINARY ADD, UNNORMALIZED

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B11 abc 000 001 $(3001)_8$	A	B	C

FUNCTION

The Floating Binary Add, Unnormalized instruction performs an algebraic binary addition of the contents of the location specified by the A-address field and the contents of the location specified by the B-address field, producing as a result an unnormalized single-precision, floating-point binary number in C; the system hunts for the next sequencing counter in demand. If C is inactive, the result is not delivered to main memory (it is retained in the accumulator) and hunting is inhibited.

A 4-bit shift to the right occurs if necessary to compensate for mantissa overflow, but no compensating left shifts occur to renormalize a result with zero in the most significant mantissa digit.

MASKING

This instruction cannot masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Binary Add, Unnormalized instruction may use inactive addressing in the manner described in Table A-1, on page A-24.

FDAU | FLOATING DECIMAL ADD, UNNORMALIZED

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B11 abc 010 001 $(3021)_8$	A	B	C

FUNCTION

The Floating Decimal Add, Unnormalized instruction performs an algebraic decimal addition of the contents of the location specified by the A-address field and the contents of the location specified by the B-address field, producing as a result an unnormalized, single-precision, floating-point decimal number in C; the system hunts for the next sequencing counter in demand. If C is inactive, the result is not delivered to main memory (it is retained in the accumulator) and hunting is inhibited.

A 4-bit shift to the right occurs if necessary to compensate for mantissa overflow, but no compensating left shifts occur to renormalize a result with zero in the most significant mantissa digit.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Decimal Add, Unnormalized instruction may use inactive addressing in the manner described in Table A-1, on page A-24.

FBSU | FLOATING BINARY SUBTRACT, UNNORMALIZED

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B11 abc 001 001 $(3011)_8$	A	B	C

FUNCTION

The Floating Binary Subtract, Unnormalized instruction performs an algebraic binary subtraction of the contents of the location specified by the B-address field from the contents of the location specified by the A-address field, producing as a result an unnormalized, single-precision, floating-point binary number in C; the system hunts for the next sequencing counter in demand. If C is inactive, the result is not delivered to main memory (it is retained in the accumulator) and hunting is inhibited.

A 4-bit shift to the right occurs if necessary to compensate for mantissa overflow, but no compensating left shifts occur to renormalize a result with zero in the most significant mantissa digit.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:

	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Binary Subtract, Unnormalized instruction may use inactive addressing in the manner described in Table A-1, on page A-24.

FDSU | FLOATING DECIMAL SUBTRACT, UNNORMALIZED

FORMAT

1 – 12	13 – 24	25 – 36	37 – 48
Command Code	Address Field	Address Field	Address Field
B11 abc 011 001 $(3031)_8$	A	B	C

FUNCTION

The Floating Decimal Subtract, Unnormalized instruction performs an algebraic decimal subtraction of the contents of the location specified by the B-address field from the contents of the location specified by the A-address field, producing as a result an unnormalized, single-precision, floating-point decimal number in C; the system hunts for the next sequencing counter in demand. If C is inactive, the result is not delivered to main memory (it is retained in the accumulator) and hunting is inhibited.

A 4-bit shift to the right occurs if necessary to compensate for mantissa overflow, but no compensating left shifts occur to renormalize a result with zero in the most significant mantissa digit.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Decimal Subtract, Unnormalized instruction may use inactive addressing in the manner described in Table A-1, on page A-24.

FBM	FLOATING BINARY MULTIPLY

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B11 abc 000 101 $(3005)_8$	A	B	C

FUNCTION

The Floating Binary Multiply instruction multiplies the contents of the location specified in A by the contents of the location specified in B. The result is a normalized, double-precision, floating-point binary number in the accumulator and the low-order product register. If the C address is active, the high-order result is delivered to the location specified and the system hunts for the next sequencing counter in demand. If C is inactive, no result is delivered to main memory (the high-order and low-order parts are retained in the accumulator and the low-order product register, respectively) and hunting is inhibited. Provision is made for automatic handling of exponential overflow and underflow in the high-order result in the following manner: If exponential underflow occurs, take the next instruction from (UTR) + 12 or (UTR) + 13. If exponential overflow occurs, take the next instruction from (UTR) + 14 or (UTR) + 15. If either occurs, store the instructions in (UTR) or (UTR) + 1. If exponential underflow occurs in the low-order result, the low-order underflow indicator is set.

A zero result in floating-point multiplication is a normalized zero (i.e., positive sign, zero mantissa, and -64 actual exponent) in both the high-order and low-order portions. Exponential overflow or underflow cannot occur with a zero result.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Binary Multiply instruction may use inactive addressing in the manner described in Table A-3, on page A-26.

FDM	FLOATING DECIMAL MULTIPLY

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B11 abc 010 101 $(3025)_8$	A	B	C

FUNCTION

The Floating Decimal Multiply instruction multiplies the contents of the location specified in A by the contents of the location specified in B. The result is a normalized, double-precision, floating-point decimal number in the accumulator and low-order product register. If the C address is active, the high-order result is delivered to the location specified and the system hunts for the next sequencing counter in demand. If C is inactive, no result is delivered to main memory (the high-order and low-order parts are retained in the accumulator and in the low-order product register respectively) and hunting is inhibited. Provision is made for automatic handling of exponential overflow and underflow in the high-order result in the following manner: If exponential underflow occurs, take the next instruction from (UTR) + 12 or (UTR) + 13. If exponential overflow occurs, take the next instruction from (UTR) + 14 or (UTR) + 15. If either occurs, store the instructions in (UTR) or (UTR) + 1. If exponential underflow occurs in the low-order result, the low-order underflow indicator is set.

A zero result in floating-point multiplication is a normalized zero (i.e., positive sign, zero mantissa, and -64 actual exponent) in both the high-order and low-order portions. Exponential overflow or underflow cannot occur with a zero result.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Decimal Multiply instruction may use inactive addressing in the manner described in Table A-3, on page A-26.

FBD	FLOATING BINARY DIVIDE

FORMAT

1 12	13 24	25 36	37 48
Command Code	Address Field	Address Field	Address Field
B01 abc 000 101 $(1005)_8$	A	B	C

FUNCTION

This instruction performs a floating binary division of the contents of B (dividend) by the contents of A (divisor), retaining the quotient in the accumulator and the remainder in the low-order product register. If the C address is active, the quotient is also delivered to the location specified and the system hunts for the next sequencing counter in demand. If C is inactive, no result is delivered to main memory (the high-order and low-order parts are retained in the accumulator and low-order product register, respectively) and hunting is inhibited. Provision is made for exponential overflow and underflow in the quotient. If exponential overflow or underflow occurs in the remainder[1] the appropriate indicator is set.

The quotient will be normalized only if the dividend is in normalized form. The remainder will not be normalized unless the dividend is zero, in which case both the quotient and the remainder will be normalized zeros and exponential overflow and underflow cannot occur. The exponent of the remainder is nine less than that of the dividend if the absolute value of the dividend mantissa equals or exceeds the absolute value of the divisor mantissa; if the absolute values of the mantissas are equal, the remainder is an unnormalized zero with an exponent nine less than that of the dividend if the absolute value of the dividend mantissa is less than the absolute value of the divisor mantissa. The sign of the remainder is the same as that of the dividend.

If the divisor is unnormalized or zero, the instruction is not performed. Instead, it is stored in U or U + 1, depending upon which sequencing counter selected it, and a division over-capacity unprogrammed transfer occurs to U + 10 or U + 11. (Note: The behavior of the system is unspecified if the program attempts to retrieve a result following division overcapacity by means of a Multiple Unload instruction.)

MASKING

This instruction cannot be masked.

[1] Since the result in the low-order product register usually has a smaller exponent than the result in the accumulator, exponential overflow in the contents of the low-order product register is normally impossible. An exception is the case of a floating-point division instruction which uses the previous contents of the accumulator as an operand, in which case it is possible to have exponential overflow in the remainder.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Binary Divide instruction may use inactive addressing in the manner described in Table 5-8, on page 5-54.

FDD	FLOATING DECIMAL DIVIDE

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B01 abc 010 101 $(1025)_8$	A	B	C

FUNCTION

This instruction performs a floating decimal division of the contents of B (dividend) by the contents of A (divisor), retaining the quotient in the accumulator and the remainder in the low-order product register. If the C address is active, the quotient is also delivered to the location specified and the system hunts for the next sequencing counter in demand. If C is inactive, no result is delivered to main memory (the high-order and low-order parts are retained in the accumulator and low-order product register, respectively) and hunting is inhibited. Provision is made for exponential overflow and underflow in the quotient. If exponential overflow or underflow occurs in the remainder, [1] the appropriate indicator is set.

The quotient will be normalized only if the dividend is in normalized form. The remainder will not be normalized unless the dividend is zero, in which case both the quotient and the remainder will be normalized zeros and exponential overflow and underflow cannot occur. The exponent of the remainder is nine less than that of the dividend if the aboslute value of the divident mantissa equals or exceeds the absolute value of the divisor mantissa; if the absolute values of the mantissas are equal, the remainder is an unnormalized zero with an exponent nine less than that of the dividend. The exponent of the remainder is ten less than that of the dividend if the absolute value of the dividend mantissa is less than the absolute value of the divisor mantissa. The sign of the remainder is the same as that of the dividend.

If the divisor is unnormalized or zero, the instruction is not performed. Instead, it is stored in U or U + 1, depending upon which sequencing counter selected it, and a division over-capacity unprogrammed transfer occurs to U + 10 or U + 11. (Note: The behavior of the system is unspecified if the program attempts to retrieve a result following division overcapacity by means of a Multiple Unload instruction.)

[1] Since the result in the low-order product register usually has a smaller exponent than the result in the accumulator, exponential overflow in the contents of the low-order product register is normally impossible. An exception is the case of a floating-point division instruction which uses the previous contents of the accumulator as an operand, in which case it is possible to have exponential overflow in the remainder.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Floating Decimal Divide instruction may use inactive addressing in the manner described in Table 5-8, on page 5-54.

FLN	NORMALIZED LESS THAN COMPARISON

FORMAT

1 12	13 24	25 36	37 48
Command Code	Address Field	Address Field	Address Field
B00 abc 011 000 $(0030)_8$	A	B	C

FUNCTION

The Normalized Less Than Comparison instruction performs a direct algebraic comparison of the contents of the locations specified by the A- and B-address fields, according to the following rules:

1. If the contents of A and B have different signs (bit 1) the operand with the positive sign exceeds the operand with the negative sign.
2. If the contents of A and B are both positive, the operand with the larger exponent exceeds the other operand. If the exponents are identical, the operand with the larger mantissa exceeds the other operand.
3. If the contents of A and B are both negative, the operand with the smaller exponent exceeds the other operand. If the exponents are identical, the operand with the smaller mantissa exceeds the other operand.

Note that this comparison is based upon the assumption of normalized operands. If either operand (or both) is unnormalized, the operand of larger magnitude may be interpreted as the smaller.

If the contents of A are less than or equal to the contents of B, the sequencing counter specified as the source of the next instruction is changed to the memory location address specified by C and hunting for the next sequencing counter in demand is inhibited. If the contents of A are greater than the contents of B, no change is made to the contents of the specified sequencing counter and hunting occurs. If C is inactive, no change is made to the contents of the specified sequencing counter and hunting is inhibited. Following the completion of a Normalized Less Than Comparison instruction, the word in the accumulator is invalid; any attempt to deliver this word to memory will therefore cause an error and will result in a call to master control.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Normalized Less Than Comparison instruction may use inactive addressing in the manner described in Table 5-16, on page 5-74.

FNN	NORMALIZED INEQUALITY COMPARISON

FORMAT

1 — 12	13 — 24	25 — 36	37 — 48
Command Code	Address Field	Address Field	Address Field
B10 abc 001 100 $(2014)_8$	A	B	C

FUNCTION

The Normalized Inequality Comparison instruction compares the 48-bit contents of the locations specified by the A- and B-address field for inequality. If the two operands are not identical, the specified sequencing counter is changed to the memory location address specified by C. If the two operands are equal, the specified sequencing counter is not changed. All properties of the Inequality Comparison, Alphabetic instruction are retained. Note that a positive zero is not equal to a negative zero. This comparison is also based on the assumption of normalized operands. If either operand (or both) is unnormalized, identical operands may be interpreted as unequal or vice versa.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Normalized Inequality Comparison instruction may use inactive addressing in the manner described in Table 5-15, on page 5-74.

FFN	FIXED-TO-FLOATING NORMALIZE

FORMAT

1	12	13	24	25	36	37	48

Command Code	Address Field	Address Field	Address Field
B00 abc 001 101 $(0015)_8$	A	B	C

FUNCTION

The Fixed-To-Floating Normalize instruction produces a normalized floating-point result based on the contents of the location specified by B and a portion of the contents of the location specified by A. The low-order 44 bits of the B operand are normalized to form the result mantissa. The sign of result is the same as the sign of the B operand; i.e., the result is positive if the B operand contains 1 or more ones in the sign bit positions. The value of bits 2 through 8 (the exponent bits) of the A operand is taken as a tentative exponent of the result. The final exponent is determined during the normalization of the mantissa according to the following rules:

1. If the low-order 44 bits of B are shifted four bits to the right to form the mantissa, the tentative exponent becomes the final exponent.
2. If the low-order 44 bits of B are not shifted at all, the tentative exponent is decreased by one to form the final exponent.
3. If the low-order 44 bits of B are shifted to the left, the tentative exponent is decreased by n to form the final exponent, where n is one greater than the number of 4-bit shifts performed.

The result is retained in the accumulator and the low-order product register, the low-order 36 bits of the low-order product register being set to zero. If the B operand is plus or minus zero, both the accumulator and the low-order product register will contain normalized zeros. If the C address is active, the high-order result is also delivered to the location specified and the system hunts for the next sequencing counter in demand. If C is inactive, no result is delivered to main memory (the high-order and low-order parts are retained in the accumulator and the low-order product register, respectively) and hunting is inhibited. Exponential underflow in the high-order result produces a normal unprogrammed transfer. If exponential underflow occurs in the low-order result, the low-order underflow indicator is set.

If the A address is inactive, the exponent of the previous contents of the accumulator is used as the A operand. Note that if the previous contents of the accumulator are in fixed-point form (e.g., the result of a fixed-point divide or a binary-to-decimal conversion), the operation will not produce a meaningful result. If the B address is inactive and the previous contents of the accumulator are in fixed-point form, these contents are used as the B operand. If the previous contents of the accumulator are in floating-point form, the sign bit in the accumulator is retained as the sign of the result, the exponent in the accumulator is ignored, and the mantissa in the accumulator is prefixed by four binary zeros to form the 44-bit operand mantissa. When this mantissa is normalized, the exponent from the A operand is reduced by one to form the result exponent.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Inactive	All	All	All

Inactive:

The Fixed-To-Floating Normalize instruction may use inactive addressing in the manner described in Table A-2, on page A-25.

ULD | MULTIPLE UNLOAD

FORMAT

1 12	13 24	25 36	37 48
Command Code	Address Field	Address Field	Address Field
B11 abc 001 101 $(3015)_8$	A	inactive (B)	C

FUNCTION

The Multiple Unload instruction does not reset the exponential overflow and underflow indicators. It transfers the contents of the accumulator to the location specified by A and the contents of the low-order product register to the location specified by C. The B address must be inactive; otherwise, the behavior of the system is unspecified. If the A address is inactive, the contents of the low-order product register are transferred to the location specified by C and the contents of the accumulator are left undefined. If the C address is inactive, the contents of the accumulator are transferred to the location specified by A and the contents of the low-order product register are placed in the accumulator. Hunting for the next sequencing counter in demand occurs only if the C address is active.

If the exponential overflow indicator is set when the Multiple Unload instruction is initiated, this instruction is followed by an unprogrammed transfer to U + 12 or U + 13. If the exponential overflow indicator is set, the unprogrammed transfer is to U + 14 or U + 15.

MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Masked	All	inactive	All

Inactive:

The Multiple Unload instruction may use inactive addressing in the manner described in Table A-4, on page A-26.

FCON	CONVERSION

FORMAT

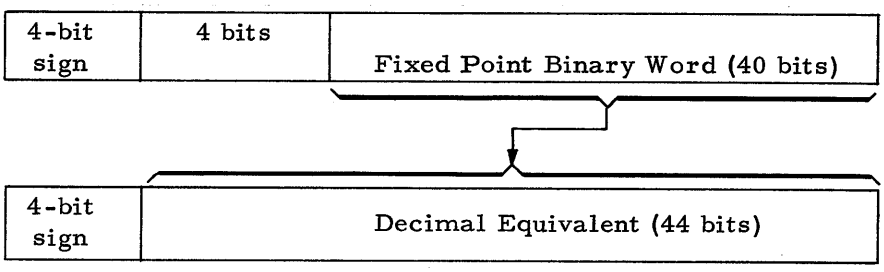
1 12	13 24	25 36	37 48
Command Code	Address Field	Address Field	Address Field
B00 abc 011 101 $(0035)_8$	A	B	C

FUNCTION

The Conversion instruction converts the contents of the location specified by B according to the contents of the location specified by A. If the B address is inactive, the previous contents of the accumulator are used as the B operand. If the A address is inactive, the behavior of the system is unspecified. Two different kinds of conversion can be specified by the value of the A operand: Fixed Decimal to Floating Binary Conversion and Floating Binary to Fixed Decimal Conversion.

1. Fixed Decimal to Floating Binary Conversion. This operation is specified by a conversion instruction in which the A operand has the octal value
   0000000000000001
   The B operand is interpreted as a signed decimal number with decimal point to the left of the high-order digit and is converted to a floating-point binary mantissa in the accumulator (unnormalized). The exponent in the accumulator is zero, while the sign is positive if the B operand contains one or more ones in the sign bit positions. The decimal remainder is retained in the low-order product register where it has the significance of a decimal fraction multiplied by 16^{-10}. If the C address is active, the result in the accumulator is delivered to the location specified and the system hunts for the next sequencing counter in demand. If C is inactive, no result is delivered to main memory and hunting is inhibited. NOTE: If the following instruction reverses the value of the high-order exponent bit, an effective fixed decimal to fixed binary conversion is obtained.

2. Floating Binary to Fixed Decimal Conversion. This operation is specified by a conversion instruction in which the A operand has the octal value
   0000000000000004
   The B operand is interpreted as a floating-point binary number and its mantissa is converted to a 44-bit fixed-point decimal fraction in the accumulator. The exponent of the B operand is ingored, while the sign of the B operand is placed in the high-order four bits of the accumulator. If the B operand mantissa is zero, however, the sign in the accumulator is a one followed by three zeros. The hexadecimal ermainder is retained in the low-order product register where it has the significance of a hexadecimal fraction multiplied by 10^{-11}. If the C address is active, the result in the accumulator is delivered to the location specified and the system hunts for the next sequencing counter in demand. If C is inactive, no result is delivered to main memory and hunting is inhibited. NOTE: This instruction can also be used to convert the low-order 40 bits of a fixed-point binary word to a 44-bit decimal equivalent.



MASKING

This instruction cannot be masked.

LEGAL ADDRESS TYPES

Active:	A	B	C
Unmasked	All	All	All

Inactive:

The Conversion instruction may use inactive addressing in the manner described in Table A-5, on page A-27.

Table A-1. Unmasked Inactive Addressing for Floating Add/Subtract Instructions

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	No operation takes place. [1]
INACTIVE	INACTIVE	ACTIVE	The contents of the accumulator are added to or subtracted from themselves, and the result is placed in the location specified by C.
INACTIVE	ACTIVE	INACTIVE	The contents of the accumulator are added to or subtracted from the contents of the location specified by B. The result is retained in the accumulator. [1]
INACTIVE	ACTIVE	ACTIVE	The contents of the accumulator are added to or subtracted from the contents of the location specified by B. The result is placed in the location specified by C.
ACTIVE	INACTIVE	INACTIVE	The contents of the location specified by A are added to or subtracted from the contents of the accumulator. The result is retained in the accumulator. [1]
ACTIVE	INACTIVE	ACTIVE	The contents of the location specified by A are added to or subtracted from the contents of the accumulator. The result is placed in the location specified by C.
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by A are added to or subtracted from the contents of the location specified by B. The result is retained in the accumulator. [1]
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by A are added to or subtracted from the contents of the location specified by B. The result is placed in the location specified by C.

[1] Hunting is inhibited.

Table A-2. Unmasked Inactive Addressing for the Floating Binary Add, Extended Precision and Floating Binary Subtract, Extended Precision Instructions

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	No operation takes place.[1]
INACTIVE	INACTIVE	ACTIVE	The contents of the accumulator are added to or subtracted from themselves. The high-order result is delivered to the location specified by C; the low-order result is placed in the low-order product register.
INACTIVE	ACTIVE	INACTIVE	The contents of the accumulator are added to or subtracted from the contents of the location specified by B. The high-order result is retained in the accumulator, and the low-order result is placed in the low-order product register.[1]
INACTIVE	ACTIVE	ACTIVE	The contents of the accumulator are added to or subtracted from the contents of the location specified by B. The high-order result is delivered to the location specified by C, and the low-order result is placed in the low-order product register.
ACTIVE	INACTIVE	INACTIVE	The contents of the location specified by A are added to or subtracted from the contents of the accumulator. The high-order result is retained in the accumulator; the low-order result is placed in the low-order product register.[1]
ACTIVE	INACTIVE	ACTIVE	The contents of the location specified by A are added to or subtracted from the contents of the accumulator. The high-order result is delivered to the location specified by C; the low-order result is placed in the low-order product register.
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by A are added to or subtracted from the contents of the location specified by B. The high-order result is retained in the accumulator, and the low-order result is placed in the low-order product register.[1]
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by A are added to or subtracted from the contents of the location specified by B. The high-order result is delivered to the location specified by C, and the low-order result is placed in the low-order product register.

[1] Hunting is inhibited.

Table A-3. Unamsked Inactive Addressing for the Floating Multiply Instructions

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	No operation takes place.[1]
INACTIVE	INACTIVE	ACTIVE	The contents of the accumulator are multiplied by themselves and the high-order product is stored in both the accumulator and C; the low-order product is stored in the low-order product register.
INACTIVE	ACTIVE	INACTIVE	The contents of the accumulator are multiplied by the contents of the location specified by B. The high-order product is stored in the accumulator, and the low-order product is stored in the low-order product register.[1]
INACTIVE	ACTIVE	ACTIVE	The contents of the accumulator are multiplied by the contents of the location specified by B. The high-order product is stored in both the accumulator and C; the low-order product is stored in the low-order product register.
ACTIVE	INACTIVE	INACTIVE	The contents of the location specified by A are multiplied by the contents of the accumulator. The high-order product is stored in the accumulator, and the low-order product is stored in the low-order product register.[1]
ACTIVE	INACTIVE	ACTIVE	The contents of the location specified by A are multiplied by the contents of the accumulator. The high-order product is stored in both the accumulator and C; the low-order product is stored in the low-order product register.
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by A are multiplied by the contents of the location specified by B. The high-order product is stored in the accumulator; the low-order product is stored in the low-order product register.[1]
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by A are multiplied by the contents of the location specified by B. The high-order product is stored in both the accumulator and C; the low-order product is stored in the low-order product register.

[1] Hunting is inhibited.

Table A-4. Unmasked Inactive Addressing for the Multiple Unload (ULD) Instruction

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	No operation takes place.[1]
INACTIVE	INACTIVE	ACTIVE	The contents of the low-order product register are transferred to the location specified by C, and the contents of the accumulator are left undefined.

Table A-4 (cont). Unmasked Inactive Addressing for the Multiple Unload (ULD) Instruction

A	B	C	Action for void addressing
INACTIVE	ACTIVE	INACTIVE	When B is active, the behavior of the system is unspecified. [1]
INACTIVE	ACTIVE	ACTIVE	When B is active, the behavior of the system is unspecified. [1]
ACTIVE	INACTIVE	INACTIVE	The contents of the accumulator are transferrred to the location specified by A, and the contents of the low-order product register are placed in the accumulator. [1]
ACTIVE	INACTIVE	ACTIVE	The contents of the accumulator are transferred to the location specified by A, and the contents of the low-order product register are transferred to the location specified by C.
ACTIVE	ACTIVE	INACTIVE	When B is active, the behavior of the system is unspecified. [1]
ACTIVE	ACTIVE	ACTIVE	When B is active, the behavior of the system is unspecified. [1]

[1] Hunting is inhibited.

Table A-5. Unmasked Inactive Addressing for the Conversion (FCON) Instruction

A	B	C	Action for void addressing
INACTIVE	INACTIVE	INACTIVE	Illegal operation code. [1]
INACTIVE	INACTIVE	ACTIVE	Illegal operation code.
INACTIVE	ACTIVE	INACTIVE	Illegal operation code. [1]
INACTIVE	ACTIVE	ACTIVE	Illegal operation code.
ACTIVE	INACTIVE	INACTIVE	The contents of the accumulator are converted according to contents of the location specified by A. [1]
ACTIVE	INACTIVE	ACTIVE	The contents of the accumulator are converted according to the contents of the location specified by A and placed in both the location specified by C and the accumulator.
ACTIVE	ACTIVE	INACTIVE	The contents of the location specified by B are converted according to the contents of the location specified by A, and the results are placed in the accumulator. [1]
ACTIVE	ACTIVE	ACTIVE	The contents of the location specified by B are converted according to the contents of the location specified by A, and the results are placed in both the location specified by C and the accumulator.

[1] Hunting is inhibited.

APPENDIX B

CHECKING FEATURES

In the Model 8200, reliability is a foremost consideration throughout every aspect of design and engineering. In addition, an extensive checking system is incorporated which will immediately sense any failure in data storage, data transmission, or control.

WORD PROCESSOR

Upon detection of a malfunction, the word processor issues a master control call. If the malfunction is detected while master control is directing operations (which occurs only in the no-hunt mode), an unprogrammed transfer (UPT) to U $\pm$ 4 and U $\pm$ 5 takes place. Certain types of main memory parity and arithmetic overflows are exceptions to the above and are listed with the individual descriptions of the checking features. The following paragraphs list and describe the malfunctions or errors which are detected.

Noninstalled Memory

Any instruction which attempts to address a main memory location above the maximum physical location of the system is detected and not completed; it results in the issuance of a master control call.

Origination of this type of malfunction is due to either a programming error or a machine malfunction. It should be noted that if the program attempts to address a location beyond 2^{23} (word-oriented, base-added address), the address registers go "end around," the address specifies the lower part of memory, and no noninstalled memory violation occurs.

Barricade Violation

Any attempt by a program group to address a main memory location, or control memory, to which that particular group has not been granted access, results in a master control call before the instruction is terminated. This type of malfunction is caused mainly by program errors; however, it may result from machine malfunctions as well.

Mod-3 Check

When an operand is brought into the arithmetic unit, the hardware generates a modulo-3 check (2 bits). The value of this two-bit digit (a value from 0 to 3) is the remainder which results when the decimal equivalent of the 48 bits is divided by 3.

When the operation has been completed, a mod-3 sum is generated based on the results of the arithmetic operation. The latter should agree with the mod-3 sums generated on the individual operands. If they do not agree, a master control call is issued.

This type of malfunction can generally be attributed to a hardware failure, however, a program attempting to do a decimal add and using hexadecimal numbers above 9 may produce a mod-3 error.

Invalid Command Code

Any group attempting to execute an instruction with an invalid command code results in a master control call. Invalid command codes are classified as follows.

1. Privileged command codes - those command codes of instructions to be executed by master control only, e.g., Acknowledge, Execute, etc.
2. Uninstalled command codes - those command codes belonging to options which are not installed.
3. Undefined command codes - all other invalid command codes.

This type of malfunction is caused mainly by a program error; however, it may result from a hardware failure as well.

Memory Local Register Check

Any failure in the parity of a word extracted from main memory will produce:

1. a master control call if the word is an instruction, or
2. an unprogrammed transfer to $U \pm 2$ and $U \pm 3$ if the word is data.

This type of malfunction is caused mainly by a hardware failure; however, certain instructions can force a bad parity.

Memory Address Register Check

Parity is generated on the addresses sent from the address generator to the memory address register (MAR). The MAR contents are checked for the correct parity and if a failure results, a master control call is issued. The instruction being executed is not completed; however, the memory cycle is completed in spite of an error. This type of malfunction can only be caused by a hardware failure.

Read Only Memory (ROM) Input Parity Check

The address of a ROM location has a parity bit associated with it which is compared with a

bit stored in the respective location after extraction. A failure produces a master control call without completion of the instruction. This type of malfunction is caused by a hardware failure.

Read Only Memory (ROM) Output Sense Amplifiers Check

After being sensed, the individual bits of the contents of a ROM location are checked upon extraction. A failure (of the sense amplifiers, not the ROM location contents itself) causes a master control call. The failure also results in the noncompletion of the instruction being executed. This type of malfunction is caused by a hardware failure only.

Control Memory Parity

In all accesses to control registers, the contents are checked for parity failures. The only exceptions to this are the read address/write address counters which are checked for parity failures by the I/O controller. If a failure occurs, the instruction is completed and a master control call is issued.

Timer-Detected Failures

The word processor is provided with a timer which detects excessive time in the execution of a single instruction or excessive time of a program group in a no-hunt loop. If an excessive length of time occurs in either, a master control call is issued. In the case of an instruction timeout, the instruction is not completed; however, when a no-hunt loop timeout occurs, the interrupt to master control takes place upon completion of the instruction being executed.

Additional Checking Features

Features are provided to perform all the unprogrammed transfers, as specified previously in Section V under the discussion of "Control Registers" (see Table 5-3). The only exception is the unprogrammed transfer for 3/4-inch tape to U $\pm$ 4 and U $\pm$ 5 (beginning-or end-of-tape). Master control will simulate these unprogrammed transfers if necessary.

VLF PROCESSOR

All detected malfunctions cause a master control call with the exception of those that produce an internal interrupt when storage protection is in effect. The following checking features are provided in the VLF processor.

Invalid Operation Codes

Any attempt to execute an instruction with an invalid operation code results in a master control call (except when storage protection is in effect). Invalid operation codes have the following classifications:

1. Uninstalled operation codes - operation codes belonging to uninstalled options.
2. Undefined operation codes - all other invalid operation codes.

This type of malfunction, which is similar to the one in the word processor, is mainly caused by a program error; however, it may result from a hardware failure as well.

Barricade Violation

A distinction must be established between the lock-and-key-type violations, which produce a master control call, and storage protection, which produces an internal interrupt. Note that when in effect, storage protection assumes priority over lock-and-key violations.

If this is the case, and a program attempts to address a main memory location with a key other than the one defined to have access, a lock-and-key-type violation will not occur because storage protection does not allow access to memory. However, if the access to main memory is attempted when storage protection is not in effect, a master control call takes place.

Noninstalled Memory

Any attempt to address a location in main memory above the maximum physical location of the system results in either the issuance of a master control call or an internal interrupt (when storage protection is in effect). The registers go "end around" according to the address mode of the VLF processor.

Mod-3 Check

After an operand is brought into the arithmetic unit and all the decimal values above 9 are changed to zero, the processor generates mod-3 checking bits. When the operation has ended, a mod-3 check is performed. If a failure occurs, a master control call is issued. This type of malfunction can only be caused by a hardware failure.

Read Only Memory (ROM) Check

The address to a ROM location and the sense amplifiers output are checked in the same way as in the word processor; that is, after being sensed, the individual bits of the contents of a ROM location are checked upon extraction. In addition, as a result of incrementing operations, parity is generated and checked. If a failure occurs, a master control call results and the instruction is not completed. This type of malfunction can only be caused by a hardware failure.

Parity Errors

All addresses and data sent to main memory and all data received from main memory have

parity bits associated with them. There is one parity bit per character which is generated from the six data bits and two punctuation bits. Any partial writing in a character (the changing of a punctuation bit without modifying the data bits or vice versa) changes the parity accordingly.

The A and B registers also have parity bits associated with them which are checked in some accesses to those registers. Any parity failure results in the issuance of a master control call after the completion of the instruction being executed. This type of malfunction can only be caused by a hardware failure.

Instruction Timeout

To prevent an instruction from entering an infinite extraction loop or an extensive execution loop, a timeout function is provided which allows a limit to be placed on the extraction and the execution time of any one instruction. The timeout function is reset to zero and begins timing whenever the processor begins to extract or to execute a new instruction. If the timeout Allow function is on, and the processor is in the standard mode when the time interval elapses, the instruction being extracted or executed is terminated, the timeout function goes on, and an internal interrupt is generated. This function guarantees that the monitor program will at some specified time regain control of the system. The timeout function is enabled by a timeout Allow function, which is set and reset by Store Variant and Indicators (SVI) and Restore Variant and Indicators (RVI) instructions.

APPENDIX C
SUMMARIES OF PDT AND PCB I/O CONTROL CHARACTERS

The following summaries of the PDT and PCB input/output control characters are to be used in conjunction with the descriptions of the PDT and PCB instructions in Section VI.

Table C-1. Summary of PDT I/O Control Characters

INPUT/OUTPUT OPERATION		PDT I/O CONTROL CHARACTER					
		C1 READ/WRITE CHANNEL	C2 CONTROL UNIT	C3 ADDITIONAL PARAMETERS	C4 ADDITIONAL PARAMETERS	C5 ADDITIONAL PARAMETERS	C6 ADDITIONAL PARAMETERS
CARD	READ	X X	X X	none	none	none	none
	PUNCH	X X	X X	none	none	none	none
See: Type 223 Card Reader (Order No. 504), Type 214-1 Card Punch (Order No. 451), Type 214-2 Card Reader/Punch (Order No. 432), Type 224 Card Reader/Punch (Order No. 506) or Type 227 Card Reader/Punch (Order No. 564)							
PAPER TAPE	READ	X X	X X	See Table C-2	none	none	none
	PUNCH	X X	X X	See Table C-3	none	none	none
See: Types 209/210 Paper Tape Equipment (Order No. 507)							
PRINTER	PRINT	X X	X X	See Table C-4	none	none	none
See: Type 222 Printers (Order No. 562)							
MAGNETIC TAPE 1/2-INCH	READ FORWARD	X X	X^1 X	6 D^3 (D=tape drive, 0 - 7)[6]	none	none	none
	READ REVERSE (Feature 010 or 011)	X X	X^1 X	2 D^4 (D=tape drive, 0 - 7)[6]	none	none	none
	WRITE	X X	X^2 X	2 D^5 (D=tape drive, 0 - 7)[6]	none	none	none
	SPACE FORWARD	X X	X^1 X	4 D (D=tape drive, 0 - 7)[6]	none	none	none
	BACKSPACE	X X	X^1 X	0 D (D=tape drive, 0 - 7)[6]	none	none	none
	ERASE	X X	X^2 X	0 D (D=tape drive, 0 - 7)[6]	none	none	none
See: Type 204B Series Magnetic Tape Unit (Order No. 503), Types 204B-11 and 204B-12 Magnetic Tape Units (Order No. 502), or Types 204C-13 and 204C-14 Magnetic Tape Units (Order No. 623)							
MAGNETIC TAPES 3/4-INCH	READ FORWARD	X X	X^1 X	6 D (D=tape drive, 0 - 3)	none	none	none
	READ SUPPRESSING CHANNEL	X X	X^1 X	5 D (D=tape drive, 0 - 3)	C 0 (C=channel to be suppressed)	none	none
	WRITE	X X	X^2 X	6 D (D=tape drive, 0 - 3)	none	none	none

Table C-1 (cont). Summary of PDT I/O Control Characters

	INPUT/OUTPUT OPERATION	PDT I/O CONTROL CHARACTER					
		C1 READ/WRITE CHANNEL	C2 CONTROL UNIT	C3 ADDITIONAL PARAMETERS	C4 ADDITIONAL PARAMETERS	C5 ADDITIONAL PARAMETERS	C6 ADDITIONAL PARAMETERS
MAGNETIC TAPES 3/4-INCH (Cont.)	SKIP WRITE	X X	X^2 X	4 D (D=tape drive, 0 - 3)	none	none	none
	BACKSPACE	X X	X^1 X	0 D (D=tape drive, 0 - 3)	none	none	none
See: Honeywell Series 200 Equipment Operators' Manual (Order No. 040)							
RANDOM ACCESS DRUM	SEARCH AND READ	X X	X^1 X	See Table C-5	0 T ... T T: 9-bit track address numbered 0 - 777 (octal)		SS: Sector address numbered 0 - 47 (octal)
	READ	X X	X^1 X	See Table C-5	none	none	none
	SEARCH AND WRITE	X X	X^2 X	See Table C-5	0 T ... T T: 9-bit track address numbered 0 - 777 (octal)		S S: Sector address numbered 0 - 47 (octal)
	WRITE	X X	X^2 X	See Table C-5	none	none	none
	READ ADDRESS REGISTER	X X	X^1 X	See Table C-5	none	none	none
See: Type 270 Random Access Drum and Control (Order No. 009)							
DISK	LOAD ADDRESS REGISTER	X X	X^2 X	0 4	none	none	none
	STORE ADDRESS REGISTER	X X	X^1 X	0 4	none	none	none
	WRITE INITIAL	X X	X^2 X	0 0 or 1 0 *	none	none	none
	EXTENDED WRITE INITIAL	X X	X^2 X	2 0 or 3 0 *	none	none	none
	WRITE	X X	X^2 X	0 1 or 1 1 *	none	none	none
	EXTENDED WRITE	X X	X^2 X	2 1 or 3 1 *	none	none	none
	SEARCH AND WRITE	X X	X^2 X	0 2 or 1 2 *	none	none	none
	EXTENDED SEARCH AND WRITE	X X	X^2 X	2 2 or 3 2 *	none	none	none
	SEARCH AND WRITE NEXT	X X	X^2 X	0 3 or 1 3 *	none	none	none
	EXTENDED SEARCH AND WRITE NEXT	X X	X^2 X	2 3 or 3 3 *	none	none	none
	SEARCH AND READ	X X	X^1 X	0 2 or 1 2 *	none	none	none
	EXTENDED SEARCH AND READ	X X	X^1 X	2 2 or 3 2 *	none	none	none
	SEARCH AND READ NEXT	X X	X^1 X	0 3 or 1 3 *	none	none	none
	EXTENDED SEARCH AND READ NEXT	X X	X^1 X	2 3 or 3 3 *	none	none	none
	READ INITIAL	X X	X^1 X	0 0 or 1 0 *	none	none	none

Table C-1 (cont). Summary of PDT I/O Control Characters

INPUT/OUTPUT OPERATION		PDT I/O CONTROL CHARACTER					
		C1 READ/WRITE CHANNEL	C2 CONTROL UNIT	C3 ADDITIONAL PARAMETERS	C4 ADDITIONAL PARAMETERS	C5 ADDITIONAL PARAMETERS	C6 ADDITIONAL PARAMETERS
DISK (Cont.)	EXTENDED READ INITIAL	X X	X[1] X	2 0 or 3 0 *	none	none	none
	READ	X X	X[1] X	0 1 or 1 1 *	none	none	none
	EXTENDED READ	X X	X[1] X	2 1 or 3 1 *	none	none	none
* Reading/writing is verified. See: Disk Devices and Controls (Order No. 514)							
ON-LINE ADAPTER	TRANSFER ID character to Series 200 memory.	X X	X X	4 X (X=unused)	none	none	none
	ACCEPT the H-800/1800 instruction defined in the ID register.[7]	X X	X X	0 0	none	none	none
	ACCEPT the word processor instruction defined in the ID register, and cause the word processor to branch to U+3 or U+5.[7]	X X	X X	0 4	none	none	none
	DO NOT ACCEPT the word processor instruction defined in the ID register; rather, cause the word processor program to branch to U+6 or U+7 (read or write error).[7]	X X	X X	1 U (U = any value from 1 - 7, octal)	none	none	none
	SET the device busy indicator.[7]	X X	X X	3 X (X=unused)	none	none	none
See: Model 212 On-Line Adapter (DSI-274)							
TYPE 281 SCCC	RECEIVE	X X	X[1] X	none	none	none	none
	TRANSMIT	X X	X[2] X	none	none	none	none
TIME-OF-DAY CLOCK	TRANSFER TIME TO MEMORY	X X	X X	none	none	none	none
CENTRAL PROCESSOR ADAPTER	TRANSFER DATA	X X	X X	none	none	none	none
See: Type 212-1 Central Processor Adapter (Order No. 239)							
MICR SORTER/READER	TRANSFER DATA	X X	X X	none	none	none	none
See: Type 233-2 MICR Control (Order No. 464)							
PLOTTER CONTROL	PLOT[7]	X X	X X	none	none	none	none
See: Type 234 Plotter Control (Order No. 561)							

Table C-1 (cont). Summary of PDT I/O Control Characters

INPUT/OUTPUT OPERATION		PDT I/O CONTROL CHARACTER					
		C1 READ/WRITE CHANNEL	C2 CONTROL UNIT	C3 ADDITIONAL PARAMETERS	C4 ADDITIONAL PARAMETERS	C5 ADDITIONAL PARAMETERS	C6 ADDITIONAL PARAMETERS
OPTICAL JOURNAL READER CONTROL	TRANSFER DATA	X X	X X	none	none	none	none
BILL FEED PRINTER CONTROL	PRINT	X X	X X	See Table C-4	none	none	none

See: Type 237 Bill Feed Printer Control (Order No. 194)

NOTES:
1. The high-order bit must be 1.
2. The high-order bit must be 0.
3. Odd parity is assumed. If even parity is required, the first octal character should be 7.
4. Odd parity is assumed. If even parity is required, the first octal character should be 3.
5. Odd parity and short gap are assumed. The first octal character should be 3 for even parity, short gap; 6 for odd parity, long gap; 7 for even parity, long gap.
6. D (tape drive) = 0-3 when the instruction is issued to the Type 203B-5 Tape Control. D = 0 or 1 when the instruction is issued to the Type 203C-7 Tape Control.
7. A complete plot can be executed by a single PDT instruction.

Table C-2. C3 Coding for Type 209 Paper Tape Reader

VALUE	B BIT	A BIT	8 BIT	4 BIT	3 BIT	1 BIT
1	Not used	One character per frame	Sense end of record	Check odd parity	Forward	Increment CLC
0	Not used	Two characters per frame	Do not sense end of record	Check even parity	Reverse	Decrement CLC

Table C-3. C3 Coding for Type 210 Paper Tape Punch

VALUE	B BIT	A BIT	8 BIT	4 BIT	2 BIT	1 BIT
1	Not used	One character per frame	Not used	Compute odd parity	00 = Do not punch parity 01 = Parity bit in channel six 10 = Parity bit in channel seven 11 = Parity bit in channel eight	
0	Not used	Two characters per frame	Not used	Compute even parity		

Table C-4. C3 Coding for Types 206 and 222 Printers and Type 237 Bill Feed Printer Control

Type 206 Printer		Type 222 Printers and Type 237 Control	
C3	INTERPRETATION	C3	INTERPRETATION
00nnnn	Print, then space the number of lines specified by nnnn (1 - 15).	00nnnn	Print, then space the number of lines specified by nnnn (0 - 15).
01nnnn	Print, then space to the head of the form if the end of the form is sensed; otherwise, space the number of lines specified by nnnn (1 - 15).	01nnnn	Print, then space to channel one of the format tape (HOF) if channel two of the format tape (EOF) is sensed; otherwise, space the number of lines specified by nnnn (0 - 15).
11nnnn	Do not print; space the number of lines specified by nnnn (1 - 15).	11nnnn	Do not print; space the number of lines specified by nnnn (0 - 15).
100011	Print, then space to the head of the form.	100xxx 101xxx	Print, then space to channel xxx.[1] Do not print; space to channel xxx.[1]
101111	Do not print; space to the head of the form.	000 001 010 011 100 101 110 111	Channel 3 Channel 4 Channel 5 Channel 1 (Head of form) Channel 6 Channel 7 Channel 8 Channel 1 (Head of form)[2]

[1] The basic Type 222-5 printer can only space to channel 1; i.e., xxx must be 011 or 111. However, when equipped with Feature 1036 (8-channel Vertical Format Tape), this printer can space to any of the channels listed.

[2] In the 237 control, space to head of form if Read PDT has been received.

Table C-5. C3 Coding for Type 270A Random Access Drum

VALUE	B BIT	A BIT	8 BIT	4 BIT	2 BIT	1 BIT
1	Override	Increment drum address register	This is a Read Address Register instruction	Drum file designation 0 - 7 (octal)		
0	Do not override	Do not increment drum address register	This is not a Read Address Register instruction			

Table C-6. Summary of PDT I/O Control Characters for Type 286 Multi-Channel Communication Control

	INPUT/OUTPUT OPERATION	A ADDRESS	PDT I/O CONTROL CHAR.		
			C1	C2	C3
TYPE 286-1, -2, -3 MCCC	FIRST DATA TRANSMISSION PDT	LOC (specifies "line 0" in 286)	X X	X[1] X	none
	RECEIVE DATA PDT	LOC+2 (specifies line address in 286)	X X	X[1] X	none
	TRANSMIT DATA PDT	LOC+2 (specifies line address in 286)	X X	X[2] X	none
	LINE CONTROL PDT	LOC (specifies address of line to be controlled) NOTE: The line control transmission PDT instructions are listed in Table 8-31, below.	X X	X[2] X	none
TYPES 286-4 AND -5 MESSAGE-MODE MCCC	TRANSMIT (Load/test state only)	Leftmost character of field from which data is transferred.	X X	X[2] X	Section address or line number, 00_8 - 63_8.
	RECEIVE (Load/test state only)	Leftmost character of field to which data is transferred.	X X	X[1] X	Section address or line number, 00_8 - 63_8.
	ASSIGN RWC AND LOAD SLC (Initialized or off-line state only)	Leftmost character of 5-character status field storing interrupt information.	X X	X X	none

NOTES: 1. The high-order bit must be 1.
2. The high-order bit must be 0.

Table C-7. Types 286-1 -2, -3 Line Control Instructions

CODE[1] (OCTAL)	INSTRUCTION	DESCRIPTION
10	Transmit last character	Inform the 286 that the last character has been sent from the central processor, and place the control unit in the receive mode for that line (after transmitting last character).
60	Receive clear	Reset the bits of the logic character in the 286 memory. (This instruction should be given when power is first turned on.)

Table C-7 (cont). Types 286-1, -2, -3 Line Control Instructions

CODE[1] (OCTAL)	INSTRUCTION	DESCRIPTION
30	Inhibit 285 (service request)	Turn off the interrupt capability of a line that is requesting service (either input or output).
50	Transmit idle character	Repeat the previously provided character indefinitely, without interrupts.
40	Transmit	Stop the line from repeating character and cause an interrupt.
74	Move Longitudinal Redundancy Check (LRC) Character	Move the LRC character from the LRC register to the data buffer register (Feature 087).
34	Special Strobe	Activate the special strobe line to a Type 285 adapter via the Type 286 control.

NOTE: The control code is stored in location LOC+1. (The low-order two bits of this code must be 0.)

Table C-8. Summary of PCB I/O Control Characters

	OPERATION		PCB I/O CONTROL CHARACTERS		
			C1	C2	C3 through Cn
TYPE 214-1 CARD PUNCH	Branch to A address if device busy		X X	X X	1 0
	Branch to A address if punch-check error		X X	X X	4 1
	Branch to A address if device unavailable. If available, set control unit to:	Punch Hollerith code[4]	X X	X X	2 7
		Punch special code	X X	X X	2 6
		Punch direct transcription code (feature 064)	X X	X X	2 5
		Generate busy signal if punch-check error	X X	X X	2 3
		Offset-stack cards with punch-check error	X X	X X	2 1
		Offset-stack the card currently at the punch station	X X	X X	3 1
	Turn the control allow function OFF		X X	X X	7 0
	Turn the control allow function ON		X X	X X	7 1
	Turn the control interrupt function OFF		X X	X X	7 4
	Branch to A address if the control interrupt function is ON		X X	X X	7 5
See: Type 214-1 Card Punch (Order No. 451)					
TYPE 214-2 CARD READER/PUNCH	Branch to A address if device busy		X X	X[3] X	1 0
	Branch to A address if cycle-check or punch-check error		X X	X[3] X	4 1
	Branch to A address if illegal punch		X X	X[3] X	4 2
	Branch to A address if device unavailable. If available, set control unit to:	Terminate punch-feed read operations, operate in Hollerith mode, and accept all other cards[4]	X X	X[3] X	2 7
		Read or punch special code	X X	X[3] X	2 6

Table C-8 (cont). Summary of PCB I/O Control Characters

OPERATION			PCB I/O CONTROL CHARACTERS		
			C1	C2	C3 through Cn
TYPE 214-2 CARD READER/PUNCH (cont)		Read or punch direct transcription code (Feature 064)	X X	X^3 X	2 5
		Generate busy signal if illegal punch	X X	X^3 X	2 4
		Generate busy signal if cycle-check or punch-check error	X X	X^3 X	2 3
		Offset-stack cards with illegal punches	X X	X^3 X	2 2
		Offset-stack cards with cycle-check or punch-check errors	X X	X^3 X	2 1
		Operate in punch-feed read mode	X X	X^3 X	2 0
		Offset-stack the card currently at the punch station	X X	X^3 X	3 1
	Turn the control allow function OFF		X X	X^3 X	7 0
	Turn the control allow function ON		X X	X^3 X	7 1
	Turn the control interrupt function OFF		X X	X^3 X	7 4
	Branch to A address if the control interrupt function is ON		X X	X^3 X	7 5
See: Type 214-2 Card Reader/Punch (Order No. 452)					
TYPE 223 CARD READER	Branch to A address if device busy		X X	X X	1 0
	Branch to A address if cycle-check error		X X	X X	4 1
	Branch to A address if illegal punch		X X	X X	4 2
	Branch to A address if device unavailable. If available, set control unit to:	Read Hollerith code and accept all error cards[4]	X X	X X	2 7
		Read special code	X X	X X	2 6
		Read direct transcription code (Feature 044)	X X	X X	2 5
		Offset-stack cards with cycle-check errors	X X	X X	2 1

Table C-8 (cont). Summary of PCB I/O Control Characters

OPERATION			PCB I/O CONTROL CHARACTERS		
			C1	C2	C3 through Cn
TYPE 223 CARD READER (cont)		Offset-stack cards with illegal punches	X X	X X	2 2
		Generate busy signal if cycle-check error	X X	X X	2 3
		Generate busy signal if illegal punch	X X	X X	2 4
		Offset-stack the card currently at the read station.	X X	X X	3 1
	Turn the control allow function OFF		X X	X X	7 0
	Turn the control allow function ON		X X	X X	7 1
	Turn the control interrupt function OFF		X X	X X	7 4
	Branch to A address if the control interrupt function is ON		X X	X X	7 5
See: Type 223 Card Reader (Order No. 504)					
TYPE 224 CARD READER/CARD PUNCH	Branch to A address if device busy		X X	X[3] X	1 0
	Branch to A address if echo-check or read registration errors		X X	X[3] X	4 1
	Branch to A address if illegal punch		X X	X[3] X	4 2
	Branch to A address if device unavailable. If available, set control unit to:	Terminate punch-feed read operations, operate in Hollerith mode, and accept all error cards[4]	X X	X[3] X	2 7
		Convert to special code	X X	X[3] X	2 6
		Operate in direct transcription mode (Feature 064)	X X	X[3] X	2 5
		Generate busy signal if illegal punch	X X	X[3] X	2 4
		Generate busy signal if echo-check or read registration errors	X X	X[3] X	2 3
		Reject cards with illegal punches (Feature 065)	X X	X[3] X	2 2
		Reject cards with echo-check or read registration errors (Feature 065)	X X	X[3] X	2 1

Table C-8 (cont). Summary of PCB I/O Control Characters

	OPERATION		PCB I/O CONTROL CHARACTERS C1	C2	C3 through Cn
TYPE 224 CARD READER/ CARD PUNCH (cont)		Operate in punch-feed read mode	X X	X[3] X	2 0
		Reject card presently in the punch station	X X	X[3] X	3 1
	Turn the control allow function OFF		X X	X[3] X	7 0
	Turn the control allow function ON		X X	X[3] X	7 1
	Turn the control interrupt function OFF		X X	X[3] X	7 4
	Branch to A address if the control interrupt function is ON		X X	X[3] X	7 5
See: Type 224 Card Reader/Punch (Order No. 506)					
TYPE 227 CARD READER	Branch to A address if device busy		X X	X X	1 0
	Branch to A address if hole-count error		X X	X X	4 1
	Branch to A address if illegal punch		X X	X X	4 2
	Branch to A address if device unavailable. If available, set control unit to:	Terminate punch-feed read operations (Feature 062), if applicable, operate in Hollerith mode, and accept all error cards[4]	X X	X X	2 7
		Read special code	X X	X X	2 6
		Read direct transcription code (Feature 040)	X X	X X	2 5
		Reject cards with hole-count errors	X X	X X	2 1
		Reject cards with illegal punches	X X	X X	2 2
		Generate busy signal if hole-count error	X X	X X	2 3
		Generate busy signal if illegal punch	X X	X X	2 4
		Place previously read card in middle stacker (Feature 017)	X X	X X	3 1
		Place previously read card in the read eject stacker (Feature 017-1)	X X	X X	3 2

Table C-8 (cont). Summary of PCB I/O Control Characters

	OPERATION		PCB I/O CONTROL CHARACTERS		
			C1	C2	C3 through Cn
TYPE 227 CARD READER (cont)	Turn the control allow function OFF		X X	X X	7 0
	Turn the control allow function ON		X X	X X	7 1
	Turn the control Interrupt function OFF		X X	X X	7 4
	Branch to A address if the control interrupt function is ON		X X	X X	7 5
See: Type 227 Card Reader/Punch (Order No. 564)					
TYPE 227 CARD PUNCH	Branch to A address if device busy		X X	X X	1 0
	Branch to A address if hole-count error (Feature 061)		X X	X X	4 1
	Branch to A address if device unavailable. If available, set control unit to:	Terminate punch-feed read operations (Feature 062), if applicable, and punch Hollerith code[4]	X X	X X	2 7
		Punch special code	X X	X X	2 6
		Punch direct transcription code (Feature 060)	X X	X X	2 5
		Reject cards with illegal punches (Feature 062)	X X	X X	2 2
		Reject cards with hole-count errors (Feature 061)	X X	X X	2 1
		Punch-feed read operations (Feature 062)	X X	X X	2 0
		Place previously punched card in middle stacker (Feature 017)	X X	X X	3 1
		Place previously punched card in the punch eject stacker (Feature 017-1)	X X	X X	3 2
	Turn the control allow function OFF		X X	X X	7 0
	Turn the control allow function ON		X X	X X	7 1

Table C-8 (cont). Summary of PCB I/O Control Characters

	OPERATION		PCB I/O CONTROL CHARACTERS		
			C1	C2	C3 through Cn
TYPE 227 CARD PUNCH (cont)	Turn the control interrupt function OFF		X X	X X	7 4
	Branch to A address if the control interrupt function is ON		X X	X X	7 5
See: Type 227 Card Reader/Punch (Order No. 564)					
TYPE 209 PAPER TAPE READER	Branch to A address if device busy		X X	X X	1 0
	Branch to A address if parity error		X X	X X	4 0
	Branch to A address if device unavailable. If available, set control unit to:	Rewind the tape (reverse direction)	X X	X X	3 0
		Run out the tape (forward direction)	X X	X X	3 2
	Turn the control allow function OFF		X X	X X	7 0
	Turn the control allow function ON		X X	X X	7 1
	Turn the control interrupt function OFF		X X	X X	7 4
	Branch to A address if the control interrupt function is ON		X X	X X	7 5
See: Types 209/210 Paper Tape Equipment (Order No. 507)					
TYPE 210 PAPER TAPE PUNCH	Branch to A address if device busy		X X	X X	1 0
	Branch to A address if tape-low condition is true		X X	X X	6 0
	Turn the control allow function OFF		X X	X X	7 0
	Turn the control allow function ON		X X	X X	7 1
	Turn the control interrupt function OFF		X X	X X	7 4
	Branch to A address if the control interrupt function is ON		X X	X X	7 5
See: Types 209/210 Paper Tape Equipment (Order No. 507)					
TYPE 206 PRINTER	Branch to A address if device busy		X X	X X	1 0
	Branch to A address if print error		X X	X X	4 0
See: Honeywell Series 200 Equipment Operators' Manual (Order No. 040)					

Table C-8 (cont). Summary of PCB I/O Control Characters

	OPERATION	PCB I/O CONTROL CHARACTERS		
		C1	C2	C3 through Cn
TYPE 222-1, -2, -3, -4, -5 PRINTER	Branch to A address if device busy	X X	X X	1 0
	Branch to A address if print error	X X	X X	4 0
	Branch to A address if paper is moving	X X	X X	2 0
	Branch to A address if busy or paper is moving	X X	X X	3 0
	Branch to A address if end of form	X X	X X	0 1
	Branch to A address if channel eight	X X	X X	0 2
	Turn the control allow function OFF	X X	X X	7 0
	Turn the control allow function ON	X X	X X	7 1
	Turn the control interrupt function OFF	X X	X X	7 4
	Branch to A address if the control interrupt function is ON	X X	X X	7 5
See: Type 222 Printers (Order No. 562) Note: PCB instructions with C3 characters 01, 02, 20, and 30 are not applicable to the basic 222-5 printer. However, the 222-5 equipped with Feature 1036 (8-Channel Vertical Format Tape) can perform all of the operations listed.				
MAGNETIC TAPE UNITS I/2-INCH	Rewind	X X	X^2 X	2 D (D=tape drive, 0 - 7)
	Rewind and release	X X	X^1 X	2 D (D=tape drive, 0 - 7)
	Branch to A address if read busy	X X	X^1 X	0 D (D=tape drive, 0 - 7)
	Branch to A address if write busy	X X	X^2 X	0 D (D=tape drive, 0 - 7)
	Branch to A address if read/write error	X X	X X	4 D (D=tape drive, 0 - 7)
	Branch to A address if beginning of tape	X X	X^1 X	6 D (D=tape drive, 0 - 7)

Table C-8 (cont). Summary of PCB I/O Control Characters

	OPERATION	PCB I/O CONTROL CHARACTERS		
		C1	C2	C3 through Cn
MAGNETIC TAPE UNITS 1/2-INCH (cont)	Branch to A address if end of tape	X X	X^2 X	6 D (D=tape drive, 0 - 7)
	Turn the control allow function OFF	X X	X^3 X	7 0
	Turn the control allow function ON	X X	X^3 X	7 1
	Turn the control interrupt function OFF	X X	X^3 X	7 4
	Branch to A address if the control interrupt function is ON	X X	X^3 X	7 5
See: Type 204B Series Magnetic Tape Units (Order No. 503), Types 204B-11 and 204B-12 Magnetic Tape Units (Order No. 502), or Types 204C-13 and 204C-14 Magnetic Tape Units (Order No. 623)				
Note: The Types 204B-11 and 204B-12 Magnetic Tape Units are limited to tape drive designations in the range 0-3 and are unable to execute the command "rewind and release." Tape drive designations for the Types 204C-13 and 204C-14 Magnetic Tape Units must be either 0 or 1; these units cannot execute the "record and release" command.				
MAGNETIC TAPE UNITS 3/4-INCH	Rewind	X X	X^2 X	2 D (D=tape drive, 0 - 3)
	Release	X X	X^1 X	2 D (D=tape drive, 0 - 3)
	Branch to A address if read busy	X X	X^1 X	0 D (D=tape drive, 0 - 3)
	Branch to A address if write busy	X X	X^2 X	0 D (D=tape drive, 0 - 3)
	Branch to A address if read/write error	X X	X X	4 D (D=tape drive, 0 - 3)
	Branch to A address if beginning of tape	X X	X^1 X	6 D (D=tape drive, 0 - 3)
	Branch to A address if end of tape	X X	X^2 X	6 D (D=tape drive, 0 - 3)
	Branch to A address if "long check" error is detected	X X	X^2 X	5 X (X=unused)

Table C-8 (cont). Summary of PCB I/O Control Characters

	OPERATION	PCB I/O CONTROL CHARACTERS		
		C1	C2	C3 through Cn
MAGNETIC TAPE UNITS 3/4-INCH (cont)	Turn the control allow function OFF	X X	X[3] X	7 0
	Turn the control allow function ON	X X	X[3] X	7 1
	Turn the control interrupt function OFF	X X	X[3] X	7 4
	Branch to A address if the control interrupt function is ON	X X	X[3] X	7 5
	See: Series 200 Equipment Operators' Manual (Order No. 040)			
TYPE 270A RANDOM ACCESS DRUM	Branch to A address if device busy[5]	X X	X X	0 X or 1 X (X=unused)
	Branch to A address if error indicator is on	X X	X X	4 X (X=unused)
	Turn the control allow function OFF	X X	X X	7 0
	Turn the control allow function ON	X X	X X	7 1
	Turn the control interrupt function OFF	X X	X X	7 4
	Branch to A address if the control interrupt function is ON	X X	X X	7 5
	See: Type 270A Random Access Drum and Control (Order No. 009)			
TYPE 212 ON-LINE ADAPTER	Branch to A address if device busy	X X	X[3] X	0 X or 1 X (X = unused)
	Branch to A address if data transfer is in progress	X X	X[3] X	7 X (X = unused)
	Branch to A address if error or incomple indicator is set	X X	X[3] X	4 X
	Branch to A address if parity error is stored	X X	X[3] X	5 X (X = unused)
	Branch to A address if incomplete error is stored	X X	X[3] X	6 X (X = unused)
	Place control character C4 in the ID register if data transfer is not in progress	X X	X[3] X	C3: 2 X (X = unused) C4: octal character to be placed in ID register
	Branch to A address unconditionally, and clear the ID register	X X	X[3] X	3 X (X = unused)
	See: Model 212 On-Line Adapter (DSI-274)			

Table C-8 (cont). Summary of PCB I/O Control Characters

	OPERATION		PCB I/O CONTROL CHARACTERS		
			C1	C2	C3 through Cn
DISK DEVICES	Branch to A address if specified drive is busy; otherwise, set control unit to:	Seek out the cylinder (specified by C5 and C6) in the pack (specified by C4).	X X	X^2 X	C3: 2 D (D-device address, 0-7) C4: 00 C5 and C6: 0000 to 0143 for the Type 258, 0000 to 0312 for the Type 259.
		Restore the specified drive to cylinder zero.	X X	X^1 X	3 D (D=device address, 0-7)
	Branch to A address if specified drive is not busy; otherwise, set control unit to:	Continue with the next sequential record the operation (read or write) being performed with the current record.	X X	X^1 X	7 D (D=device address, 0-7).
	Branch to A address if control busy.		X X	X^2 X	1 0
	Branch to A address if device busy.		X X	X^2 X	C3: 0 D (D=device address, 0-7) C4: 0 0 or another valid C3 character
	Branch to A address if a general exception condition occurred during the preceding PDT instruction.		X X	X^2 X	5 0
	Branch to A address if the TLR flag is set.		X X	X^2 X	6 0
	Set control unit to override setting of FORMAT WRITE PERMIT switch.		X X	X^2 X	4 0
	Turn control allow function OFF.		X X	X^2 X	7 0
	Turn control allow function ON.		X X	X^2 X	7 1
	Turn drive allow function OFF.		X X	X^2 X	7 2
	Turn drive allow function ON.		X X	X^2 X	7 3
	Turn control interrupt function OFF.[7]		X X	X^2 X	7 4

Table C-8 (cont). Summary of PCB I/O Control Characters

	OPERATION	PCB I/O CONTROL CHARACTERS		
		C1	C2	C3 through Cn
DISK DEVICES (cont)	Branch to A address if control interrupt function is ON.	X X	X^2 X	7 5
	Turn drive interrupt function OFF. Branch to A address if drive interrupt function is ON.	X X	X^2 X	7 6
See: Disk Devices and Controls (Order No. 514)				
TYPE 281 SCCC	Branch to A address if device busy	X X	X^3 X	1 0
	Branch to A address if parity error	X X	X^3 X	4 0
	Branch to A address if error other than parity error	X X	X^3 X	5 0
	Branch to A address if the 281 is in transmit mode and requesting data for transmission onto line	X X	X^3 X	6 0
	Branch to A address if the 281 is in receive mode and requesting that central processor take received data	X X	X^3 X	6 1
	Turn the allow function OFF	X X	X^6 X	7 0
	Turn the allow function ON	X X	X^6 X	7 1
	Turn the interrupt function OFF	X X	X^6 X	7 4
	Branch to A address if allow and interrupt functions are ON	X X	X^6 X	7 5
TYPE 213-3 INTERVAL TIMER	Branch to A address if device busy (Feature 071)	0 0	X^6 X	1 0
	Turn the allow function OFF	0 0	X^6 X	7 0
	Turn the allow function ON	0 0	X^6 X	7 1
	Turn the allow function ON (Feature 071)	0 0	X^6 X	7 3 (C4 - C6 specify time interval)
	Turn the interrupt function OFF	0 0	X^6 X	7 4
	Branch to A address if interrupt function is ON	0 0	X^6 X	7 5

Table C-8 (cont). Summary of PCB I/O Control Characters

	OPERATION	PCB I/O CONTROL CHARACTERS		
		C1	C2	C3 through Cn
TYPE 213-3 INTERVAL TIMER (cont)	**Turn the interrupt function OFF (Feature 071)**	0 0	X^6 X	7 6
	Branch to A address if interrupt function is ON (Feature 071)	0 0	X^6 X	7 7
See: Type 213-3 Interval Timer and Feature 071 Interval Selector (Order No. 082)				
TYPE 213-4 TIME OF DAY CLOCK	Branch to A address if device busy	X X	X X	1 0
TYPE 212-1 CENTRAL PROCESSOR ADAPTER	Branch to A address if device busy	X X	X^6 X	0 X or 1 0
	Branch to A address if device busy, and reserve	X X	X^6 X	C3: 2 0 C4: 0 0
	Branch to A address if reserve action by this central processor was not successful	X X	X^6 X	C3: 2 0 C4: 0 0 C5: 6 1
	Branch to A address if 212-1 is not set for data transfer (initiator)	X X	X^6 X	6 1
	Branch to A address if 212-1 is set for data transfer (responder)	X X	X^6 X	6 4
	Turn the allow function OFF	X X	X^6 X	7 0
	Turn the allow function ON	X X	X^6 X	7 1
	Turn the interrupt function OFF	X X	X^6 X	7 4
	Branch to A address if allow and interrupt functions are ON	X X	X^6 X	7 5
See: Type 212-1 Central Processor Adapter (Order No. 239)				
TYPE 234 PLOTTER CONTROL	Branch to A address if control unit busy.	X X	X X	1 0
	Turn the allow function OFF	X X	X X	7 0
	Turn the allow function ON	X X	X X	7 1
	Turn the interrupt function OFF	X X	X X	7 4
	Branch to A address if interrupt function is ON	X X	X X	7 5
See: Type 234 Plotter Control (Order No. 561)				

Table C-8 (cont). Summary of PCB I/O Control Characters

	OPERATION		PCB I/O CONTROL CHARACTERS		
			C1	C2	C3 through Cn
TYPE 235 OPTICAL JOURNAL READER CONTROL	Branch to A address if device busy		X X	X X	1 0
	Branch to A address if reader is not set for data transfer or if control is busy		X X	X X	0 1
	Turn the allow function OFF		X X	X X	7 0
	Turn the allow function ON		X X	X X	7 1
	Turn the interrupt function OFF		X X	X X	7 4
	Branch to A address if interrupt function is ON		X X	X X	7 5
TYPE 233-2 MICR SORTER-READER	Branch to A address if control busy		X X	X X	1 0
	Select stacker designated; Branch to A address if: 1. the sorter-reader is not ready; or 2. the 10-millisecond stacker selection period has elapsed; or 3. the leading edge of the document to be sorted has not passed the reading station; or 4. the leading edge has passed the reading station and a PDT instruction has not yet been issued; or 5. the sorter-reader is performing an automatic reject on the document in question	Stacker 0	X X	X X	2 0
		Stacker 1	X X	X X	2 1
		Stacker 2	X X	X X	2 2
		Stacker 3	X X	X X	2 3
		Stacker 4	X X	X X	2 4
		Stacker 5	X X	X X	2 5
		Stacker 6	X X	X X	2 6
		Stacker 7	X X	X X	2 7
		Stacker 8	X X	X X	3 0
		Stacker 9	X X	X X	3 1
		Stacker X	X X	X X	3 2
		Stacker Y	X X	X X	3 3
		Reject Stacker	X X	X X	3 7
	Start feed. Branch to A address if feed cannot be started due to: 1. the sorter-reader not being ready; or 2. proper restart procedures not followed		X X	X X	3 4
	Stop feed. Branch to A address if sorter-reader is not ready		X X	X X	3 5
	Set pocket-light control. Branch to A address if:		X X	X X	3 6

Table C-8 (cont). Summary of PCB I/O Control Characters

	OPERATION		PCB I/O CONTROL CHARACTERS		
			C1	C2	C3 through Cn
TYPE 233-2 MICR SORTER-READER (cont)	1. the sorter-reader is not ready; or 2. a pocket-light control PCB is already in process				
	Branch to A address if:	Amount field error	X X	X X	4 0
		Process control field error	X X	X X	4 1
		Account field error	X X	X X	4 2
		Transit field error	X X	X X	4 3
		Auxiliary on-us field error	X X	X X	4 4
		Device error	X X	X X	5 0
		Passed document condition	X X	X X	5 1
	Operate in normal mode		X X	X X	6 0
	Operate in short-document mode		X X	X X	6 1
	Branch to A address if on-us field is complete		X X	X X	6 2
	Branch to A address if last document was a control document		X X	X X	6 3
	Branch to A address if end-of-file		X X	X X	6 4
	Advance batch counter one digit. Branch to A address if the sorter-reader is not stopped or the batch counter is currently being advanced		X X	X X	6 5
	Turn allow function OFF		X X	X X	7 0
	Turn allow function ON		X X	X X	7 1
	Turn interrupt function OFF		X X	X X	7 4
	Branch to A address if interrupt function is ON		X X	X X	7 5
	See: Type 233-2 MICR Control (Order No. 464)				
TYPE 237 BILL FEED PRINTER CONTROL	Branch to A address if:	Device busy	X X	X^3 X	1 0
		Form is moving	X X	X^2 X	2 0
		Device busy or form is moving	X X	X^3 X	3 0
		Print error or read check	X X	X^3 X	4 0
		Validity error	X X	X^3 X	4 1
	Branch on channel 2 (EOF) of format tape		X X	X^2 X	0 1
	Branch on channel 8 of format tape		X X	X^2 X	0 2

Table C-8 (cont). Summary of PCB I/O Control Characters

	OPERATION	PCB I/O CONTROL CHARACTERS		
		C1	C2	C3 through Cn
TYPE 237 BILL FEED PRINTER CONTROL (cont)	Turn on validity check indicator	X X	X[1] X	2 0
	Turn the allow function OFF	X X	X X	7 0
	Turn the allow function ON	X X	X X	7 1
	Turn the interrupt function OFF	X X	X X	7 4
	Branch to A address if read interrupt function is ON	X X	X[1] X	7 5
	NOTE: The two operations "Branch to A address if form is moving" and "Turn on validity check indicator" are both specified with a C3 character of 20_8, but are distinguished by the high-order bit of C2.			

See: Type 237 Bill Feed Printer Control (Order No. 194)

[1]The high-order bit must be 1.

[2]The high-order bit must be 0.

[3]The high-order bit is set to 1 for input operations and to 0 for output operations.

[4]This control character should precede all other control characters that set the control to perform a certain action. It is the programmer's responsibility to set the control to the desired mode of operation at the beginning of the run.

[5]As the drum control does not permit reading from one drum file while writing on another, it is considered busy if either a read or a write operation is in progress. (The value of the high-order bit in C2 is thus immaterial in this case.)

[6]The high-order bit is ignored.

[7]The interrupt functions of both the control and the disk device are automatically turned on when a "not busy" status is reached by the control or disk device, respectively.

Table C-9. Summary of PCB I/O Control Characters for Type 286 Multi-Channel Communication Control

	OPERATION	PCB I/O CONTROL CHARACTERS			
		C1	C2	C3	C4 through Cn
TYPE 286-1, -2 -3 MCCC	Branch to A address if device busy. If not busy, set the 286 to stop scanning and continue the program in sequence	X X	X X	1 0	none
	Turn the allow function OFF	X X	X X	7 0	none
	Turn the allow function ON	X X	X X	7 1	none
	Branch to A address if the interrupt was due to the 286 requesting service	X X	X X	7 5	none

Table C-9 (cont). Summary of PCB I/O Control Characters for Type 286 Multi-Channel Communication Control

	OPERATION	PCB I/O CONTROL CHARACTERS			
		C1	C2	C3	C4 through Cn
TYPE 286-4, -5 MESSAGE-MODE MCCC	Branch to A address if device busy	X X	X X	1 0	none
	Branch to A address if parity error	X X	X X	4 0	none
	Branch to A address if the interrupt was due to the 286 requesting service	X X	X X	7 5	none
	Turn the allow function ON	X X	X X	7 1	none
	Turn the allow function OFF	X X	X X	7 0	none
	Set the 286 to the load/test state	X X	X X	2 5	none
	Provide line orientation for load/test operation	X X	X X	4 1	none
	Turn the load/test state and line orientation OFF	X X	X X	2 4	none
	Turn the interrupt function OFF	X X	X X	7 4	none
	Release the RWC(s) assigned to the 286	X X	X X	2 7	none
	Set the halt/continue indicator to halt	X X	X X	2 0	none
	Set the halt/continue indicator to continue	X X	X X	2 1	none
	Turn the parity error indicator and the parity error interrupt function OFF	X X	X X	2 6	none
	Request the address of the next transfer that is to take place from the line designated by C4, and branch to the A address	X X	X X	3 6	C4: 00 to 77
	Abort the present instruction to the line designated by C4, generate an interrupt, initiate the next instruction to the same line, and branch to the A address	X X	X X	3 3	C4: 00 to 77
	Abort the present instruction to the line designated by C4, initiate the next instruction to the same line, and branch to the A address	X X	X X	3 2	C4: 00 to 77
	Reset synchronization for the line designated by C4, and branch to the A address	X X	X X	3 7	C4: 00 to 77
	Activate the special strobe line to the 285 adapter designated by C4, and branch to the A address	X X	X X	3 4	C4: 00 to 77
	Deliver to the 286 the information specified by C5 et seq. for the next instruction to the line designated by C4, and branch to the A address	X X	X X	3 0	C4: 00 to 77 (See Table C-10 for C5 et seq.)
	Deliver to the 286 the information specified by C5 et seq. for the next instruction to the line designated by C4; then abort the present instruction to that line, generate an interrupt, initiate the next instruction to the same line, and branch to the A address	X X	X X	3 3	C4: 00 to 77 (See Table C-10 for C5 et seq.)
	Deliver to the 286 the information specified by C5 et seq. for the next instruction to the line designated by C4, then abort the present instruction to that line, initiate the next instruction to the same line, and branch to the A address	X X	X X	3 2	C4: 00 to 77 (See Table C-10 for C5 et seq.)

APPENDIX D
MISCELLANEOUS TABLES

Table D-1. Control Register Designations

CONTROL REGISTER	VARIANT CHARACTER (LCR & SCR Instructions)
CLC7	00
CLC1	01
CLC2	02
CLC3	03
CLC7'	04
CLC1'	05
CLC2'	06
CLC3'	07
SLC7	10
SLC1	11
SLC2	12
SLC3	13
SLC7'	14
SLC1'	15
SLC2'	16
SLC3'	17
CLC8	20
CLC4	21
CLC5	22
CLC6	23
CLC8'	24
CLC4'	25
CLC5'	26
CLC6'	27
SLC8	30
SLC4	31
SLC5	32
SLC6	33
SLC8'	34

Table D-1 (cont). Control Register Designations

CONTROL REGISTER	VARIANT CHARACTER (LCR & SCR Instructions)
SLC4'	35
SLC5'	36
SLC6'	37
AC0	--
AC1	--
AC2	--
AC3	--
CSR	64
EIR	66
AAR	67
BAR	70
IIR	76
SR	77

Table D-2. Extended Move (EXM) Conditions

CONDITIONS	VARIANT BITS V_6 V_5 V_4 V_3 V_2 V_1
Type of Move	
1. A-field data bits ⟶ B	X X X X X 1
2. A-field word-mark bits ⟶ B	X X X X 1 X
3. A-field item-mark bits ⟶ B	X X X 1 X X
Direction of Move	
1. right to left	X X 0 X X X
2. left to right	X X 1 X X X
Termination of Move	
1. automatic after single-character move	0 0 X X X X
2. A-field word mark	0 1 X X X X
3. A-field item mark	1 0 X X X X
4. A-field record mark	1 1 X X X X

Table D-3. Branch on Condition Test (BCT) SENSE Switch Conditions

VARIANT CHARACTER (Octal)	BRANCH CONDITION
00	Unconditional
01	SENSE Switch 1 On
02	SENSE Switch 2 On
03	SENSE Switches 1 and 2 On
04	SENSE Switch 3 On
05	SENSE Switches 1 and 3 On
06	SENSE Switches 2 and 3 On
07	SENSE Switches 1, 2, and 3 On
10	SENSE Switch 4 On
11	SENSE Switches 1 and 4 On
12	SENSE Switches 2 and 4 On
13	SENSE Switches 1, 2, and 4 On
14	SENSE Switches 3 and 4 On
15	SENSE Switches 1, 3, and 4 On
16	SENSE Switches 2, 3, and 4 On
17	SENSE Switches 1, 2, 3, and 4 On
20	Unconditional
21	SENSE Switch 5 On
22	SENSE Switch 6 On
23	SENSE Switches 5 and 6 On
24	SENSE Switch 7 On
25	SENSE Switches 5 and 7 On
26	SENSE Switches 6 and 7 On
27	SENSE Switches 5, 6, and 7 On
30	SENSE Switch 8 On
31	SENSE Switches 5 and 8 On
32	SENSE Switches 6 and 8 On
33	SENSE Switches 5, 6, and 8 On
34	SENSE Switches 7 and 8 On
35	SENSE Switches 5, 7, and 8 On
36	SENSE Switches 6, 7, and 8 On
37	SENSE Switches 5, 6, 7, and 8 On

NOTE: When testing for a multiple SENSE switch condition, a branch occurs only if all of the specified conditions are met.

Table D-4. Branch on Condition Test (BCT) Indicator Conditions

VARIANT CHARACTER (Octal)	BRANCH CONDITION
40	Do not branch
41	B<A (Low Compare)
42	B=A (Equal Compare)
43	B≤A (Low or Equal Compare)
44	B>A (High Compare)
45	B≠A (Unequal Compare)
46	B≥A (High or Equal Compare)
47	Unconditional
50	Overflow
51	Overflow or B<A
52	Overflow or B=A
53	Overflow or B≤A
54	Overflow or B>A
55	Overflow or B≠A
56	Overflow or B≥A
57	Unconditional
60	Zero Balance
61	Zero Balance or B<A
62	Zero Balance or B=A
63	Zero Balance or B≤A
64	Zero Balance or B>A
65	Zero Balance or B≠A
66	Zero Balance or B≥A
67	Unconditional
70	Overflow or Zero Balance
71	Overflow or Zero Balance or B<A
72	Overflow or Zero Balance or B=A
73	Overflow or Zero Balance or B≤A
74	Overflow or Zero Balance or B>A
75	Overflow or Zero Balance or B≠A
76	Overflow or Zero Balance or B≥A
77	Unconditional

NOTE: When testing for a multiple indicator condition, a branch occurs if any one of the specified conditions is met.

Table D-5. Branch on Character Condition (BCC) Conditions

VARIANT CHARACTER (Octal)	BRANCH CONDITION
00	Unconditional
01	A bit is 1
02	B bit is 1
03	B and A bits are 11
04	B and A bits are 00
05	B and A bits are 01 (Positive sign)
06	B and A bits are 10 (Negative sign)
07	B and A bits are 11 (same as 03)
10	Word-mark bit is 1
11	Word-mark bit is 1, A bit is 1
12	Word-mark bit is 1, B bit is 1
13	Word-mark bit is 1, B and A bits are 11
14	Word-mark bit is 1, B and A bits are 00
15	Word-mark bit is 1, Positive sign
16	Word-mark bit is 1, Negative sign
17	Word-mark bit is 1, B and A bits are 11
20	Item-mark bit is 1
21	Item-mark bit is 1, A bit is 1
22	Item-mark bit is 1, B bit is 1
23	Item-mark bit is 1, B and A bits are 11
24	Item-mark bit is 1, B and A bits are 00
25	Item-mark bit is 1, Positive Sign
26	Item-mark bit is 1, Negative Sign
27	Item-mark bit is 1, B and A bits are 11
30	Record mark
31	Record mark, A bit is 1
32	Record mark, B bit is 1
33	Record mark, B and A bits are 11
34	Record mark, B and A bits are 00
35	Record mark, Positive sign
36	Record mark, Negative sign
37	Record mark, B and A bits are 11
40	No punctuation (Word-mark and-Record mark bits are 00)
41	No punctuation, A bit is 1
42	No punctuation, B bit is 1
43	No punctuation, B and A bits are 11
44	No punctuation, B and A bits are 00
45	No punctuation, Positive sign
46	No punctuation, Negative sign
47	No punctuation, B and A bits are 11
50	Word mark
51	Word mark, A bit is 1
52	Word mark, B bit is 1
53	Word mark, B and A bits are 11
54	Word mark, B and A bits are 00
55	Word mark, Positive sign
56	Word mark, Negative sign
57	Word mark, B and A bits are 11

Table D-5 (cont). Branch on Character Condition (BCC) Conditions

VARIANT CHARACTER (Octal)	BRANCH CONDITION
60	Item mark
61	Item mark, A bit is 1
62	Item mark, B bit is 1
63	Item mark, B and A bits are 11
64	Item mark, B and A bits are 00
65	Item mark, Positive sign
66	Item mark, Negative sign
67	Item mark, B and A bits are 11
70	Unconditional
71	Word mark or A bit is 1
72	Word mark or B bit is 1
73	Word mark or B and A bits are 11
74	Word mark or B and A bits are 00
75	Word mark or Positive sign
76	Word mark or Negative sign
77	Word mark or B and A bits are 11

APPENDIX E
INTERRUPT PROCESSING

The execution of main-program instructions by the VLF processor can be interrupted by an external interrupt source and by an internal interrupt source.

EXTERNAL INTERRUPT

An external interrupt signal can be generated by any one of three sources:

1. The operator's console;
2. The Monitor Call instruction; or
3. A peripheral control.

The first two sources interrupt the processor directly: in the case of the console, the operator simply presses the INTERRUPT button; the Monitor Call instruction interrupts the processor when it is executed. However, a peripheral control interrupts program sequence as directed by the settings of two programmable storage functions contained within the control, as described on page E-5.

The interrupt signal sets indicators to show the source (whether 1., 2., or 3., above) and the type (external) of interruption. These indicators can be stored and then tested by programmed instruction as described further in this appendix. The processor acts upon the interrupt signal when the following conditions are present:

1. The processor is in the RUN mode (i.e., the processor is executing, without manual intervention, stored-program instructions under control of SR).
2. The processor is not in the external interrupt mode.
3. An instruction op code is about to be extracted.
4. A memory cycle is allocated to the processor.

It should be noted that condition 3. above does not cause an extensive delay if the VLF processor is attempting to extract a Peripheral Data Transfer (PDT) instruction and the specified read/write channel or peripheral control is "busy." The attempt to issue a PDT instruction to a busy read/write channel or peripheral control does not "stall" the central processor. Rather, the instruction is "re-extracted": SR is set back to the address of the PDT op code, so that condition 3. recurs immediately after the channel or control is found busy.

When the VLF processor is interrupted, it performs the following functions:

1. Stores the current status of the arithmetic, comparison, address mode, and trap mode indicators in the auxiliary indicators register (AIR).
2. Clears the arithmetic indicators.
3. Enters the 3-character, non-trap mode.
4. Interchanges the contents of SR and EIR and branches to the instruction whose op code address was previously stored in EIR.
5. Enters the external interrupt mode.

The interrupt signal is maintained until one of the following steps is taken.

1. A PDT instruction is issued to the peripheral control.
2. The Interrupt function for the peripheral control is turned off.
3. The central processor is initialized.

INTERNAL INTERRUPT

An internal interrupt signal, caused by a "violation" of storage protection, is generated by the VLF processor. Processor indicators are set by the internal interrupt signal to show the cause (e.g., op code violation) and the type (internal) of interruption. These indicators can be stored and then tested by programmed instruction as described further in this appendix.

The processor reacts to the internal interrupt signal when the conditions described on page E-1 are present (i.e., the processor is in the RUN mode, is not in the external interrupt mode, is about to extract an op code, and is presently allocated a memory cycle) plus one additional condition: the processor must not only not be in the external interrupt mode but also must not be in the internal interrupt mode. Thus, the following levels of interrupt priority exist in the VLF processor.

1. If the processor is in the noninterrupt (standard) mode, normal program sequence can be interrupted by either an external or an internal source.
2. If the processor is in the internal interrupt mode, program sequence can be interrupted only by an external interrupt source.
3. If the processor is in the external interrupt mode, program sequence can not be interrupted.[1]

The VLF processor responds to an internal interrupt signal as follows:

1. The contents of SR and IIR are interchanged, and the program branches to the instruction whose op code address was previously stored in IIR.

[1] Interrupt signals generated by any or all of the three external sources (peripheral control, console, or Monitor Call instruction) may continue to occur while the processor is in the external interrupt mode. The priority in which the interrupts are accommodated is determined by the program (i.e., according to the programmer-established sequence of interrupt source tests).

2. The processor enters the internal interrupt mode.

Note that the status of the arithmetic, comparison, address mode, and trap mode indicators are not stored in AIR automatically when the processor responds to an internal interrupt signal. The storing (and subsequent restoring) of the contents of these indicators is the responsibility of the internal interrupt program.

INTERRUPT PROGRAMMING

Three of the four interrupt control instructions perform basic functions in an interrupt routine:

1. The Store Variant and Indicators instruction (SVI) stores two types of information: (a) information which must be preserved for subsequent return to the interrupted program (e.g., indicator settings, variant register contents, etc.); and (b) information required to identify the interrupt source.
2. The Restore Variant and Indicators instruction (RVI) restores the pertinent information stored by the SVI instruction before returning to the interrupted program.
3. The Resume Normal Mode instruction (RNM) returns the processor to continue sequencing in the interrupted program, unless the sector bits of SR have been modified.

The fourth interrupt control instruction — Monitor Call (MC) — causes an external interruption and, therefore, is not coded in the interrupt routine itself.

Other instructions are required in the interrupt routine to store and exercise control over address register contents, as shown in Figures E-1 and E-2. The interrupt routines in these figures are assumed to be executed in the same sector as the interrupted program; if not, or if interrupt processing modifies the sector bits in SR, the appropriate sector bits must be stored upon entering the routine and restored when exiting.

For proper re-entry to the interrupted program, the same set of indicators stored by the SVI instruction should be restored by the RVI. Since the RVI instruction prepares the processor to re-enter the interrupted program, it should be followed immediately by the RNM. Note that the A- and B-address register settings at the time of the interrupt should also be restored before re-entering the interrupted program. The external interrupt coding shown in Figure E-1 exploits the ability to restore the address registers automatically by storing their contents in the address fields of the RNM instruction. This technique requires that variant bit V_2 of the RVI instruction (see page 6-94) be a zero in order to ensure that the RNM is executed in the maximum address mode of the machine. In an internal interrupt routine, on the other hand, the indicators associated with bit V_2 must be stored and restored by the SVI/RVI instructions. Therefore, since the address mode of executing the RNM may not be maximum, the address fields of this instruction must not be coded. Instead, the address register settings must be stored in memory and restored by means of LCR instructions, as shown in Figure E-2.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

	CARD NUMBER	TYPE	WAR	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1				AAR	CEQU	#1C67	A-ADDRESS REGISTER
2				BAR	CEQU	#1C70	B-ADDRESS REGISTER
3				MAX	CEQU	#1C60	MAXIMUM ADD. MODE FOR C.P. IS 4
4				ALLS	CEQU	#1C75	INDICATORS STORED
5				ALLR	CEQU	#1C25	INDICATORS RESTORED
6					ADMODE	MAX	SET MAXIMUM ADDRESSING MODE
7				RESTOR	RVI	ENTER+2,ALLR	RESTORE INDICATORS
8				EXIT	RNM	Ø,Ø	EXIT WITH AAR & BAR RESTORED
9				ENTER	SVI	ALLS	ENTER AND STORE INDICATORS
10					DCW	#5	RESERVE STORAGE FOR INDICATORS
11					CAM	MAX	ENTER MAXIMUM ADDRESS MODE
12					SCR	EXIT+4,AAR	SAVE AAR
13					SCR	EXIT+8,BAR	SAVE BAR
14							
15							
16							EXTERNAL
17							INTERRUPT
18							ROUTINE
19							
20					B	RESTOR	BRANCH TO RESTOR AND EXIT

Figure E-1. Sample Coding for External Interrupt Routine

The first example (see Figure E-1) shows the initial and final coding to be used in an external interrupt routine. It is assumed that the address of the location tagged ENTER was previously stored in EIR, so that the presence of an external interrupt signal results in the automatic branch to the location tagged ENTER. It is assumed that the 4-character addressing mode is the maximum addressing mode of the processor for which this routine is written.

NOTE: If the interrupt routine is not in the maximum addressing mode prior to branching to the location tagged RESTOR, a Change Addressing Mode instruction — CAM/MAX — must precede the RVI instruction so that the complete contents of any necessary control memory locations may be restored.

Figure E-2 shows the initial and final coding written for an internal interrupt routine. It is assumed that the address of the location tagged START was previously stored in IIR and that the maximum addressing mode of the processor is the 4-character mode.

The initial and concluding instructions in an internal interrupt routine are similar to those in an external interrupt routine, except that the SVI instruction must store the indicators associated with bit V_2 and must not store the contents of the auxiliary indicators register (AIR). All other pertinent indicators are stored by the SVI instruction and are subsequently restored by the RVI instruction at the conclusion of the routine.

CODING FORM

PROBLEM _____ PROGRAMMER _____ DATE _____ PAGE ___ OF ___

	CARD NUMBER	TYPE	MARK	LOCATION	OPERATION CODE	OPERANDS	
	1 2 3 4 5	6	7	8 14	15 20	21 62	63 80
1				AAR	CEQU	#1C67 A-ADDRESS REGISTER	
2				BAR	CEQU	#1C7Ø B-ADDRESS REGISTER	
3				MAX	CEQU	#1C6Ø MAXIMUM ADD. MODE FOR CP IS 4	
4				INDS	CEQU	#1C73 ALL BUT AIR INDICATORS	
5				INDR	CEQU	#1C33 ALL BUT AIR AND INT. INDICATORS	
6				SAVEA	DCW	#4C TEMPORARY STORAGE FOR AAR	
7				SAVEB	DCW	#4C TEMPORARY STORAGE FOR BAR	
8					ADMODE	4 SET MAXIMUM ADDRESSING MODE	
9				RESTOR	LCR	SAVEA,AAR RESTORE AAR	
10					LCR	SAVEB,BAR RESTORE BAR	
11					RVI	START+2,INDR RESTORE ALL BUT AIR AND INT IND.	
12					RNM	EXIT	
13				START	SVI	INDS ENTER AND STORE ALL BUT AIR IND.	
14					DCW	#5 STORAGE FOR ALL BUT AIR IND.	
15					CAM	MAX ENTER MAXIMUM ADDRESSING MODE	
16					SCR	SAVEA,AAR SAVE AAR	
17					SCR	SAVEB,BAR SAVE BAR	
18							
19							
20						INTERNAL	
21						INTERRUPT ROUTINE	
22							
23							
24					B	RESTOR BRANCH TO RESTOR AND EXIT	

Figure E-2. Sample Coding for Internal Interrupt Routine

PERIPHERAL CONTROL INTERRUPT

Generally, a peripheral control's interrupt facility includes two interrelated functions: the Allow function and the Interrupt function. Certain controls have more than one set of functions (e.g., two sets for Disk Pack controls, but one set for magnetic tape controls). When a peripheral control becomes ready to accept a PDT instruction (i.e., reaches a "not-busy" status), it transmits a signal to turn on the Interrupt function, but this signal must be complemented by one from the Allow function (turned on by a PCB instruction) in order to complete the interrupt signal for transmission to the processor (see Figure E-3).[1] When the Interrupt function is turned on, the interrupt signal is repeated continuously until the VLF processor is interrupted or the signal is turned off.

[1] This activity does not apply where there is no Allow function.

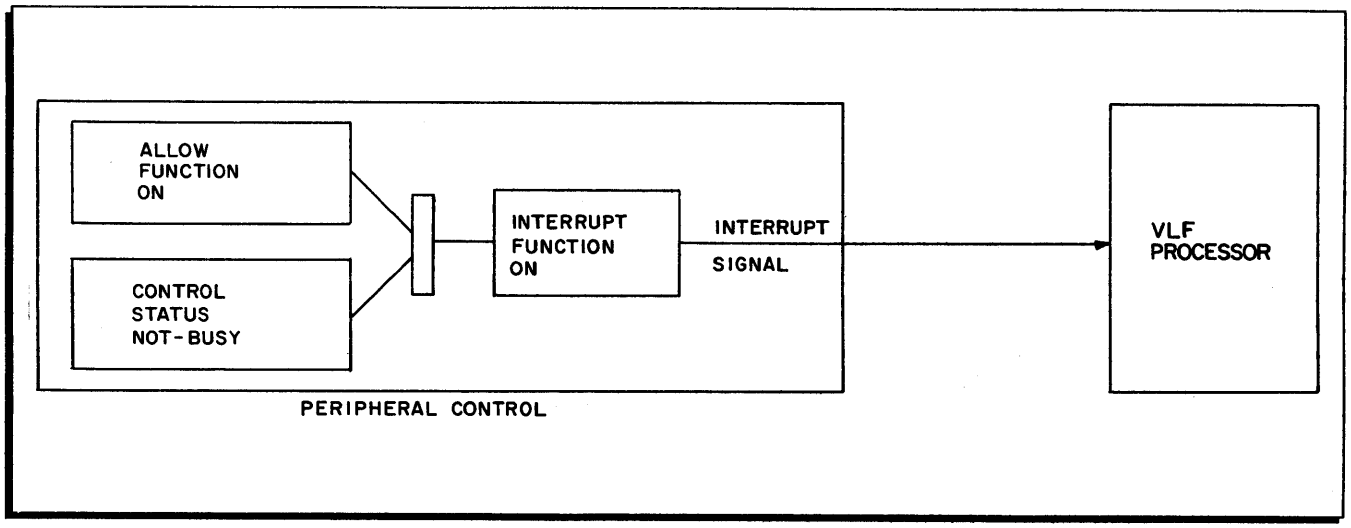


Figure E-3. Interrupt Signal Generated by Peripheral Control

The interrupt facility for a peripheral control can be activated or deactivated simply by turning the Allow function on or off, respectively. If the Allow function is off at the time the peripheral control becomes not busy and all error information is stored, the interrupt signal can be neither completed nor transmitted. Another method of inhibiting the interrupt facility is to turn off the Interrupt function; this function will not be turned on again until the control completes another PDT instruction. Note that if an interrupt has occurred and the Allow function has then been turned off, the Allow function should not be turned on again until either the Interrupt function has been turned off or a PDT instruction has been initiated by the control; otherwise, an interrupt occurs immediately.

There are various methods of turning the Allow or Interrupt function on or off. The Allow function can be turned on or off by a PCB instruction; similarly, the Interrupt function can be tested or turned off by a PCB instruction. Also, when the peripheral control receives a PDT instruction, its Interrupt function is turned off automatically; at completion of the PDT, a pulse is sent to turn on the Interrupt function. In any situation, both functions are turned off by initializing the central processor.

Specific PCB C3 characters for individual controls are listed in Tables C-7 through C-9. The C3 character in a PCB instruction may be used either to control or to test the status of a peripheral control's interrupt facility. The general formats of the C3 characters relating to interrupt control and test are:

1110x0 - Turn off the Allow function.

1110x1 - Turn on the Allow function.

1111x0 - Turn off the Interrupt function.

1111x1 - Branch to A if the Interrupt function is on.

The 2-bit, shown here as x, is normally zero if the control being addressed contains only one set of Interrupt/Allow functions. If two sets of functions are present, this bit is set to identify the particular set being tested or controlled. All of these C3 characters result in a branch to A if the device addressed is not operable. Table E-1 summarizes Interrupt/Allow control and test operations for most peripheral controls; exceptions are noted in individual device manuals.

More than one control character can be used to specify multiple control and/or test operations in a PCB instruction. However, care must be taken in the use of certain combinations of these characters. For example, it is entirely possible for an interrupt to occur between extractions of control characters. In such a case, if control characters for "Branch on Interrupt" and "Turn Off Interrupt" were specified (in that order), the Interrupt function might be turned off without being acknowledged.

Table E-1. Summary of Interrupt/Allow Function Control and Test Operations

CONTROL/TEST OPERATIONS	RESULTING EFFECTS	
	ALLOW FUNCTION	INTERRUPT FUNCTION
Manual		
INITIALIZE Button	Turned off	Turned off
Program - PCB Control Char.[1]		
70	Turned off	None
71	Turned on	None
74	None	Turned off
75	None	Branch to A if on
Peripheral Control		
Upon receipt of PDT	None	Turned off
When PDT completed	None	Turned on if Allow on

[1] All of these PCB control characters will result in a branch to A if the device addressed is not operable.

APPENDIX F

STORAGE PROTECTION AND BASE RELOCATION

Storage protection with base relocation in the VLF processor places a barrier above and below the area of memory where the active program is to operate. When storage protection and base relocation are in effect (i.e., the protect indicator and the base relocation indicator are on), an active program is restricted to operating within the area of memory defined by the two barriers. The active program's area of memory is defined as follows:

1. The programmer sets the lower boundary of the area with the Load Index/Barricade Register (LIB) instruction, specifying the absolute memory address which is an integral multiple of 4,096 characters. The LIB instruction places this number (the absolute address) in the base relocation register. The lower boundary of the active program's area is the leftmost (lowest) core storage location within this bank.
2. The upper boundary of the active program's area can be set as in 1, above. The LIB instruction is used to specify the length of the active program which must be an integral multiple of 4,096 characters. This number (length of program) is placed in the index/barricade register by the LIB instruction. The index/barricade register will then provide the upper boundary for the active program's area of operation.

In order to put storage protection and base relocation into effect, the following conditions must be present:

1. The programmer must have turned on the protect and the base relocation indicators by issuing a Restore Variant Indicators (RVI) instruction specifying the two indicators (see page 6-128).
2. The processor must be in the standard (noninterrupt) mode.

INDEX REGISTERS

When storage protection and base relocation are in effect, each core resident program has its own set of 15 index registers which is located in the leftmost sixty locations of the 4,096 character banks specified by the current contents of the base relocation register.

VLF PROCESSOR MODES

The VLF processor can operate in any one of three modes:

1. The standard mode.
2. The external interrupt mode (see Appendix D), or
3. The internal interrupt mode.

Internal Interrupt

When storage protection is in effect (i.e., the protect indicator is on and the processor is operating in the standard mode), certain operations are defined as violations of that protection. These violations are discussed below. A violation causes a violation indicator to be set which, in turn, causes an internal interrupt to occur at the next opportunity. The "next opportunity" means that moment when all of the following conditions are present:

1. The processor is in the run mode (i.e., automatically executing stored program instructions under the control of the sequence register).
2. The processor is about to extract an op code,
3. A memory cycle is allocated to the processor,
4. The processor is in the standard mode (i.e., not in external or internal interrupt mode), and
5. No peripheral or console interrupt signal is being received.

When an internal interrupt occurs, the contents of the sequence register and the internal interrupt register are interchanged and the VLF processor enters the internal interrupt mode. The status of the processor indicators are not stored automatically; therefore, the programmer must perform this function with a Store Variant and Indicators (SVI) instruction. The SVI instruction also clears the violation indicator so that an internal interrupt will not occur when a return is made to the standard mode. While in the internal interrupt mode, any external interrupt will cause the processor to switch to the external interrupt mode.

If an external interrupt occurs while the processor is in the internal mode, bit 1 of the character stored by V5 of the SVI instruction indicates the condition. If it is desired to revert to the standard rather than the internal interrupt mode after servicing the external interrupt, this bit should be changed to zero before executing the RVI instruction.

Note that three basic differences exist between the external interrupt mode and the internal interrupt mode:

1. A unique control memory location, the internal interrupt register (IIR), contains the address of the subroutine which services the internal interrupt.
2. The processor is subject to being interrupted by an external interrupt while still in the internal interrupt mode, but the reverse is not true.
3. No processor indicators are stored or altered (the address mode is not changed) upon entering the internal interrupt mode.

VIOLATIONS OF STORAGE PROTECTION

The following operations, which constitute violations of storage protection, fall into two general categories: address violations and op code violations.

1. An attempt to transfer information internally to memory locations outside the active program area. However, no violation occurs when information is transferred internally from outside of the active program area. An internal transfer violation is detected when all of the following conditions are present:

   a. The bank and sector bits in the A- or B-address register following instruction extraction are equal to or greater than the corresponding bits stored in the index/barricade register,

   b. A location outside of the active program's area is addressed as the result location.

   c. The protect and the base relocation indicators are on.

   d. The program in control is operating in the standard mode, and

   e. The instruction is not a PDT.

   The above conditions are checked as the instruction is being executed. If all of these conditions are met, the internal interrupt address violation indicator is set, and the instruction proceeds to normal completion except that no information is transferred into memory (i.e., the write cycle is inhibited). The next opportunity for the internal interrupt to occur is at the extraction of the next op code. After the internal interrupt mode is entered, the internal interrupt register contains the address of the op code following the instruction which caused the violation, and the A- and B-address registers continue to increment or decrement, as appropriate.

2. An attempt to extract a PDT instruction (input or output) whose effective A address specifies a memory location outside of the active program's area. Since the PDT instruction is one of the operations normally prohibited when storage protection is in effect (see 4., below), the proceed indicator must be set in order for the instruction to be extracted beyond the op code. Assuming that the proceed indicator is set, the starting address of the PDT operation is examined for address violation. Once it is determined that the effective A address specifies an address which is outside of the active program's area, no operation is performed (i.e., the specified read/write channel is not tested and the specified peripheral control is not addressed), the internal interrupt address violation indicator is set, the sequence register is advanced to the next op code, and an internal interrupt occurs.

   Note that a PDT instruction is checked for possible violation during the extraction phase, while a nonperipheral instruction is checked during its execution phase (see 1., above). If a PDT instruction passes this test during extraction, it is free to be executed and thereby cause data to be transferred. If the information being transferred extends out of the active program's area, no address violation is detected. To insure that this will not occur, the user must set a record mark immediately prior to the upper boundary of the active program's area.

   As mentioned previously, storage protection (and the checking functions related to it) are in effect only when the processor is operating in the standard mode. However, attempted violations of the protection by PDT

instructions executed in either of the two interrupt modes can be detected if the proceed indicator is set on.

3. An attempt to read from a main memory location whose address is greater than the main memory capacity actually present in the machine but within the addressing capacity of the memory address register. Such an addressing attempt results in a parity error which normally causes the machine to halt. The storage protect hardware assumes that out-of-range addressing has been attempted, no halt occurs, nor is data transferred; instead, the internal interrupt address violation indicator is set, instruction execution is prematurely terminated, and an internal interrupt occurs.

4. An attempt to execute a privileged op code. A privileged op code is one which is (a) not defined for the VLF processor; (b) not recognized by the VLF processor; or (c) prohibited when storage protection is in effect. The privileged op codes in category (c) are:

   a. H (Halt)
   b. LCR (Load Control Registers)
   c. PDT (Peripheral Data Transfer)
   d. PCB (Peripheral Control and Branch)
   e. SVI (Store Variant and Indicators)
   f. RVI (Restore Variant and Indicators)
   g. RNM (Resume Normal Mode)
   h. LIB (Load Index/Barricade Register)

   The above op codes are "privileged" in the sense they are allowed to be executed in either of the interrupt modes but are prohibited in the standard mode while storage protection is in effect (one exception to this is discussed under "Proceed Indicator" below). Such op codes are categorized by the fact that they could possibly alter the monitor's knowledge of the status of the system or cause some action which is intolerable under certain conditions (e.g., a halt during transfer of data from a communications device). Since an undefined op code or one which is not installed on the processor would normally cause a halt due to a program check, such usage has the same effect as that of a privileged op code.

   NOTE: Op code "00" is defined as an Internal Interrupt Call, and falls within the category of privileged op codes.

   If a privileged op code is extracted when storage protection is in effect, the op code violation indicator is turned on, the sequence register is set back to the location of the op code, the operation is terminated, and an internal interrupt occurs. Once the internal interrupt mode is entered, the programmer has two choices: (1) if he wishes to execute the privileged instruction, he must set the proceed indicator and issue a Resume Normal Mode (RNM) command;[1] (2) if he wishes to bypass the privileged instruction, he must set the internal interrupt register to the location of the next sequencial op code and issue a Resume Normal Mode instruction.[2]

[1] The instruction will still not be executed if it involves an address violation.

[2] If the internal interrupt register is not advanced to the next op code, the return to normal mode results in the privileged op code again being extracted, thus causing an endless loop.

PROCEED INDICATOR

The proceed indicator can be turned on by the Restore Variant and Indicators (RVI) instruction. Turning this indicator on permits the execution of one privileged instruction in the standard mode without op code checking or item-mark trapping being performed. The indicator is turned off following the extraction of any op code in the standard mode. It can also be turned off in either of the interrupt modes by a Store Variant and Indicators (SVI) instruction.

The proceed indicator can also be used to force the checking of the A address of a PDT instruction executed ineither the internal or the external interrupt mode. Thus, turning on this indicator prior to the extraction of a PDT instruction in a nonstandard (interrupt) mode results in the same address violation check as though it were extracted in the standard mode with storage protection in effect. If the effective A address is found to specify a location outside the active program's area, the actions described below are performed.

1. When the violation occurs in the internal interrupt mode:
   a. The internal interrupt address violation indicator is set.
   b. Further extraction of the instruction is not performed and the sequence register is set to the location of the next sequential op code.
   c. An internal interrupt does not occur since theprocessor is already in the internal interrupt mode. Instead, the condition of the internal interrupt address violation indicator must be tested by the programmer after he has stored the status of the indicator via an SVI instruction. The SVI instruction also clears the indicator so that it will not cause an internal interrupt to occur when the standard mode is entered later.
2. When the violation occurs in the external interrupt mode:
   a. The external interrupt address violation indicator is set.
   b. Further extraction of the instruction is not performed and the sequence register is set to the location of the next sequential op code.
   c. An internal interrupt does not occur since this is impossible while in the external interrupt mode. Instead, the condition of the external interrupt address violation indicator must be tested by the programmer according to the method described in 1.c., above.

APPENDIX G

INSTRUCTION TIMING TABLES

Table G-1. Instruction Timing of Variable-Length-Field Processing Subsystem

Name of Operation	Standard Format	Execution Time, Microseconds[1] Addresses Not Indexed	Addresses Indexed
Fixed-Point Arithmetic			
Decimal Add[2]	A/A, B	11.15	15.45
Decimal Subtract[2]	S/A, B	11.15	15.45
Decimal Multiply[3]	M/A, B	119.1	123.4
Decimal Divide[3]	D/A, B	59.5	63.8
Binary Add	BA/A, B	11.15	15.45
Binary Subtract	BS/A, B	11.15	15.45
Zero and Add	ZA/A, B	9.65	13.95
Zero and Subtract	ZS/A, B	9.65	13.95
Logical Functions			
Half-Add	HA/A, B	10.85	15.15
Extract	EXT/A, B	10.85	15.15
Compare	C/A, B	9.6	13.9
Substitute	SST/A, B, V	7.0	11.3
Branch	B/A	4.1	7.4
Branch on Condition Test	BCT/A, V	4.3	7.6
Branch on Character Condition	BCC/A, B, V	5.8	10.1
Branch if Character Equal	BCE/A, B, V	5.8	10.1
Branch on Bit Equal	BBE/A, B, V	5.8	10.1
General Control Functions			
Set Word Mark	SW/A, B	5.3	9.6
Set Item Mark	SI/A, B	5.3	9.6
Clear Word Mark	CW/A, B	5.3	9.6
Clear Item Mark	CI/A, B	5.3	9.6
Halt	H/A	4.5	7.8
No Operation	NOP	3.4	—
Resume Normal Mode	RNM/A, B	4.5	8.8
Store Variant and Indicators	SVI/V	8.4	—
Restore Variant and Indicators	RVI/A, V	7.1	10.4
Monitor Call	MC	3.4	—
Store Control Registers	SCR/A, V	5.7	9.0
Load Control Registers	LCR/A, V	5.7	9.0
Change Addressing Mode	CAM/V	3.5	—
Change Sequencing Mode	CSM/A, B, V	4.3	8.6
Store Index/Memory Protect Indicators	SIB/A	5.3	8.6
Load Index/Memory Protect Indicators	LIB/A	5.7	9.0
Data Move Instructions			
Move Characters to Word Mark	MCW/A, B	9.25	13.55
Load Characters to A-Field Word Mark	LCA/A, B	9.25	13.55
Move Item and Translate	MIT/A, B, V^1, V^2 V^3	14.8	19.1
Move and Translate	MAT/A, B, V^1, V^2	14.1	18.4
Extended Move	EXM/A, B, V	8.8	13.10
Editing			
Move Characters and Edit[4]	MCE/A, B	13.15	17.45
Input/Output			
Peripheral Data Transfer	PDT/A, $C^1, C^2, \ldots C_n$	-	—
Peripheral Control and Branch	PCB/A, $C^1C^2, \ldots C_n$	-	—

[1] The execution times given in this table are based on realistic situations involving the three-character addressing mode. The data fields referenced by both the A and B addresses are five characters long. Times for indexed operations are based on indexing of all address fields. In actual practice, higher speeds will be attained because in many cases abbreviated instruction formats can be used.

[2] Based on no recomplement cycle; if recomplement cycle required, add 3.8 microseconds.

[3] Average time.

[4] Based on four characters scanned in both second and third passes.

Table G-2. Instruction Timing of Word Processing Subsystem

NAME OF OPERATION	OPERAND SIZE	EXECUTION TIME IN MICROSECONDS[4]				
			APPROXIMATE MEAN TIME WITH INTERLEAVING			
		MINIMUM	4-WAY	2-WAY	1-WAY	MAXIMUM
Fixed-Point Arithmetic						
Binary Add, Subtract	44 Bits + Sign	1.75	2.19	2.48	3.0	3.0
Binary Accumulate	N Words	1.25 + .25N	1.35 + .25N	1.5 + .5N	1.5 + .75N	1.5 + .75N
Binary Multiply	44 Bits + Sign	5.25	5.25	5.25	5.25	5.25
Binary Divide	44 Bits + Sign	14.0	14.0	14.0	14.0	14.0
Decimal Add, Subtract	11 Digits + Sign	1.75	2.19	3.0	3.0	3.0
Decimal Accumulate	N Words	1.25 + .25N	1.35 + .25N	1.5 + .5N	1.5 + .75N	1.5 + .75N
Decimal Multiply	11 Digits + Sign	6.0	6.0	6.0	6.0	6.0
Decimal Divide	11 Digits + Sign	14.0	14.0	14.0	14.0	14.0
Word Add, Difference	48 Bits	1.75	2.19	2.48	3.0	3.0
Scientific Processing Instructions[1]						
Floating Binary Add, Subtract	—	2.25	3.0	3.3	4.15	9.25
Floating Binary Add, Subtract — Extended	—	2.75	3.5	4.5	5.5	10.5
Floating Binary Add, Subtract — Unnormalized	—	2.25	3.0	3.3	4.15	9.0
Floating Binary Multiply	—	7.5	7.5	7.5	7.5	7.5
Floating Binary Divide	—	14.0	14.0	14.0	14.0	14.0
Floating Decimal Add, Subtract	—	2.25	3.0	3.3	4.15	9.25
Floating Decimal Add, Subtract — Unnormalized	—	2.25	3.0	3.3	4.15	9.0
Floating Decimal Multiply	—	7.75	7.75	7.75	7.75	7.75
Floating Decimal Divide	—	14.0	14.0	14.0	14.0	14.0
Normalized Less Than Comparison	—	3.0	3.25	3.25	3.25	3.5
Normalized Inequality Comparison	—	3.0	3.25	3.25	3.25	3.5
Multiple Unload	—	1.75	2.19	2.48	3.0	3.0
Fixed-Decimal to Floating-Binary Conversion	—	14.25	14.25	14.25	14.25	14.25
Floating-Binary to Fixed-Decimal Conversion	—	10.25	10.25	10.25	10.25	10.25
Fixed-to-Floating Normalize	—	1.75	2.19	2.48	3.0	3.0
Logical Functions						
Half Add	48 Bits	1.75	2.19	2.48	3.0	3.0
Superimpose	48 Bits	1.75	2.19	2.48	3.0	3.0
Substitute	48 Bits	2.5	2.94	3.23	3.75	3.75
Extract	48 Bits	1.75	2.19	2.48	3.0	3.0
Inequality Comparison, Alphanumeric	48 Bits	3.0	3.25	3.25	3.25	3.5
Inequality Comparison, Numeric	11 Digits or 44 Bits + Sign	3.0	3.25	3.25	3.25	3.5
Less Than Or Equal to Comparison, Alphanumeric	48 Bits	3.0	3.25	3.25	3.25	3.5
Less Than Or Equal to Comparison, Numeric	11 Digits or 44 Bits + Sign	3.0	3.25	3.25	3.25	3.5
Shift Instructions[2]						
Shift Word and Substitute	48 Bits	1.75	2.95	3.26	3.77	4.5
Shift Preserving Sign and Substitute	11 Digits or 44 Bits + Sign	1.75	2.95	3.26	3.77	4.5
Shift Word and Extract	48 Bits	1.75	3.46	3.86	4.13	4.5
Shift Preserving Sign and Extract	11 Digits or 44 Bits + Sign	1.75	3.46	3.86	4.13	4.5
Shift Word and Select	48 Bits	3.0	4.50	5.22	5.58	6.0
Data Move Instructions						
Transfer A to C	48 Bits	1.75	2.1	2.15	2.25	2.25
Transfer A to B and Go to C	48 Bits	1.75	2.1	2.15	2.25	2.25
Multiple Transfer	N Occurrences of Same Word	1.25 + .75N	1.25 + .95N	1.25 + 1.15N	1.25 + 1.3N	1.25 + 1.5N
N-Word Transfer	N Words	1.25 + .75N	1.25 + .95N	1.25 + 1.15N	1.25 + 1.3N	1.25 + 1.5N
Item Transfer	N Words	1.25 + .75N	1.25 + .95N	1.25 + 1.15N	1.25 + 1.3N	1.25 + 1.5N
Record Transfer	N Words	1.25 + .75N	1.25 + .95N	1.25 + 1.15N	1.25 + 1.3N	1.25 + 1.5N
General Control Functions						
Compute Orthocount	N Words	3.75 + 1N	3.75 + 1N	5.25 + 1N	6 + 1.25N	6 + 1.5N
Check Memory Parity	48 Bits	2.5	2.62	2.77	3.0	3.0
Multiprogram Control	—	3.5	3.50	3.50	3.75	3.75
Proceed	—	1.75	1.75	1.75	1.75	1.75
Simulator	—	1.75	2.62	2.77	3.0	3.0
Input/Output and Other Peripheral Functions[3]						
Peripheral Control and Branch	—	—	—	—	—	—
Peripheral Control and Transfer	—	—	—	—	—	—

[1] Single-precision, floating-point operands consist of a 1-bit sign, followed by a 7-bit exponent and a 40-bit mantissa.

[2] Times for additions and subtractions are based on no equalization.

[3] Execution times for shift instructions are based on an average shift distribution over 1-48 bits.

[4] Trapping of input/output operations into the master control facility requires approximately 4 microseconds; thereafter, the time required depends upon the operation being performed.

[5] Minimum times are for maximum overlap with 4-way interleaving, all addresses active and direct main memory. Maximum times are for all addresses indexed-indirect with no memory overlap. Mean times are calculated on a weighted distribution of addressing types and assuming random distribution of operands among the available memory banks. All times are exclusive of masking, which can take a maximum of two additional memory cycles, depending on overlap.

HONEYWELL

TECHNICAL PUBLICATIONS REMARKS FORM

TITLE: MODEL 8200
HARDWARE REFERENCE MANUAL

DATED: AUGUST, 1967

FILE NO: 113.0011.0000.0-685

ERRORS NOTED IN PUBLICATION:

Fold

Cut Along Line

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION:

Fold

(Please Print)

FROM: NAME _____ DATE _____

COMPANY _____

TITLE _____

ADDRESS _____

_____

FIRST CLASS
PERMIT NO. 39531
NEWTON HIGHLANDS
MASS.

BUSINESS REPLY MAIL

No postage stamp necessary if mailed in the United States

POSTAGE WILL BE PAID BY

HONEYWELL
151 NEEDHAM STREET
NEWTON HIGHLANDS, MASS. 02161

ATT'N: MARKETING INFORMATION SERVICES, MS 251

Cut Line

Honeywell