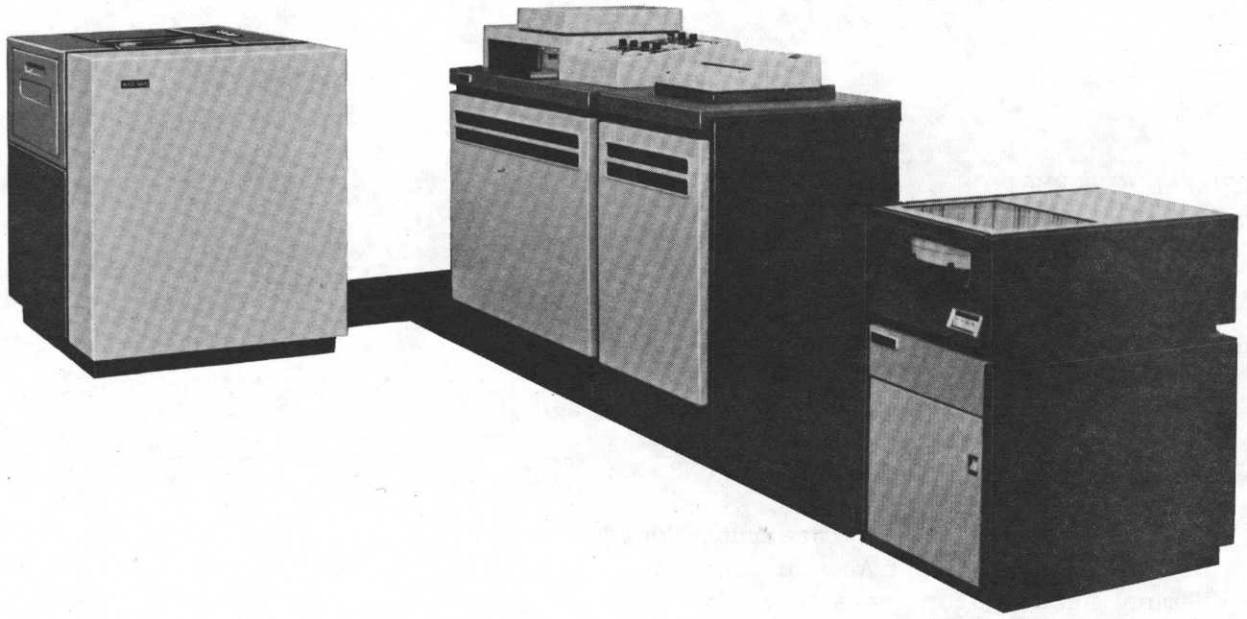
NCR CENTURY 101 PROCESSOR



This publication contains a functional description of the NCR Century 101 Processor. The Introduction section highlights the features of the processor; subsequent chapters (Memory, Arithmetic Logic Unit, I/O Control, Options, and Specifications) cover the features and their related functions in more detail.

This publication is not intended for use as a programming or operating manual and should not be used as such.

TABLE OF CONTENTS

PAGE

INTRODUCTION

SYSTEM DESCRIPTION 5
PROCESSOR DESCRIPTION 5
Memory 6
Arithmetic Logic Unit 6
I/O Control 7
Operator's Console 7

MEMORY

GENERAL INFORMATION 9
DATA REPRESENTATION 9
Interpreting a Bye of Data 9
NCR Century Codes (ASCII) 10
Data Fields 11
ADDRESSING A FIELD IN MEMORY 13
Addressing Memory By Console Entry 14
Addressing Memory By Hardware Command 17
Special Addressing Modes 19
AREAS RESERVED FOR HARDWARE FUNCTIONS 24
Index Register Words 26
ME, PE/ICC and Program Interrupt Control Areas 27
Divisor and Dividend Accumulators 27
Memory Accumulator 28
Control Words 28

ARITHMETIC LOGIC UNIT

GENERAL INFORMATION 31
ALU CONTROL SECTION 31
Processor Timing 31
Normal Control Logic 31
Miscellaneous Decision Logic 32
HARDWARE REGISTERS 32
FLAGS AND INDICATORS 33
Comparison Flags (L, E, and G) 34
Overflow Flag (OF) 34
Repeat Indicator (RI) 35
Interrupt Permit Indicator (IP) 35
Interrupt Indicator (II) 36
Memory Error Indicator (ME) 36

TABLE OF CONTENTS (Continued)

PAGE

Program Error Indicator (PE) 36
Illegal Command Code Indicator (ICC) 37
Error Indicator (EI) 37
Read-Only-Memory Error Indicator (RE) 37
ALU ADDER 38
Using the Adder for Command Setup 39
Arithmetic Adder Functions 41
Nonarithmetic Adder Functions 52
FUNCTIONAL OPERATION OF THE ENTIRE ALU SECTION 61
Command Format 61
Command Setup 64
Command Execution 69
System Malfunctions 70
Between Commands Testing 70
Trapping Flows 71

I/O CONTROL

GENERAL INFORMATION 75
PERIPHERAL TYPES 77
Integrated Peripherals 77
Freestanding Peripherals 77
PERIPHERAL TRANSFER RATES 78
TRUNK BANDWIDTH 79
TRUNK AND PERIPHERAL PRIORITY 79
Trunk 0 Priority Network 79
I/O Priority Scheme 80
SYSTEM I/O BANDWIDTH 81
I/O FUNCTIONAL OPERATION 82
Peripheral Selection 82
Data Transfer 84
Termination of the I/O Operation 86

OPTIONS

GENERAL INFORMATION 92
EXTENDED MEMORY 92
I/O TRUNKS 1 AND 6 92
50-HERTZ ELECTRICAL CAPABILITIES 92
REMOTE ALARM 92
INTEGRATED CARD OR TAPE READER (COT) 92
682-101 Integrated Card Reader 93
662-100 Integrated Punched Tape Reader 93

TABLE OF CONTENTS (Continued)

	PAGE
PRINTERS	93
I/O WRITER	96
CRT I/O CONSOLE	97
TOD CLOCK	97
HARDWARE MULTIPLY/DIVIDE COMMANDS	98
Hardware MULTIPLY Command.	98
Hardware DIVIDE Command.	99
HARDWARE LOGIC COMMAND	102
ONLINE COMMUNICATIONS	102
MULTIPROGRAMMING FEATURE	103

INTRODUCTION

SYSTEM DESCRIPTION

The NCR Century 101 is a powerful data processing system that can have a configuration to meet a wide range of business and scientific requirements. It performs well in batch processing applications where low-cost file capability is a major consideration. It also offers modest real-time capability, common trunk interfacing for a wide range of peripherals, low-cost simultaneity, and upward compatability.

The modular concept of the central processing unit permits the user to expand his system without changing processors or software. The processor's memory can be expanded in increments of 8K bytes from 16K to 32K, then in 16K increments from 32K, to 48K, to 64K. I/O capabilities can be expanded from the basic 2-trunk configuration (1 low-speed trunk and 1 high-speed trunk) to a 4-trunk configuration (2 low-speed trunks and 2 high-speed trunks). If online data communication with remote terminals or other processing systems is required, communication capabilities can be added.

A wide range of NCR Century peripherals can be used with this processor. Printers with speeds from 300 lines per minute up to 2000 lpm are available. Freestanding disc units with transfer rates ranging from 312 KB to 500 KB and storage capacities of 4.9 megabytes to 48 megabytes per disc pack are available for use with the NCR Century 101.

PROCESSOR DESCRIPTION

The central processing unit (CPU) is the heart of the NCR Century 101 System. A single cabinet (1) houses the entire processor memory, (2) contains all of the logic necessary for data manipulation and I/O control, and (3) contains the necessary logic and electronics for both the operator's console and the integrated COT (card or tape) reader. The auxiliary cabinet (shown in the photo on the previous page) must be included if the system configuration contains an integrated printer or I/O Writer. This cabinet contains the power supply and logic for the printer and also houses the standard I/O Writer or the thermal I/O Writer if desired.

The CPU has a basic repertoire of 34 hardware commands. Three optional commands are also available (LOGIC, MULTIPLY, and DIVIDE). Hardware commands perform many of the functions normally performed by slower software routines in smaller systems, thereby providing greater processing efficiency. For example, the signed versions of the add and subtract commands enable the system to manipulate packed data without forcing the programmer to first unpack it and then repack it following manipulation; also, the test character commands enable the system to compare two 8-bit characters and (depending upon the result) transfer control in the same operation.

The processor consists of four major sections: the memory section, the arithmetic logic unit, the I/O control section, and the operator's console.

Memory

The processor's memory, which is contained entirely within the processor's main cabinet, is binarily addressable by command or console entry. The basic storage unit is the byte, which consists of eight data bits and one parity bit. For faster command setup and addressing capabilities, memory is accessed two bytes at a time; however, data may be written into memory in either one- or two-byte increments. The time to access or store a byte of information in memory (memory cycle time) is 1.2 microseconds.

The processor logic provides four addressing modes: direct addressing (Mode 0), indirect addressing (Mode 1), indirect addressing (Mode 2), and incremental addressing (Mode 3). These four addressing modes provide the programmer with a very versatile method of memory access. He may specify an absolute address with Mode 0 addressing, use the address specified in another area of memory or another command by using Mode 1 or 2 addressing, and incrementally step through an area of memory using Mode 3 addressing.

Each addressing mode permits the use of index registers to create an effective memory address. There are 63 index registers in the processor's memory. Registers 1 through 12 are reserved for software control; the remaining 51 may be used as work registers.

Arithmetic Logic Unit

The processor's arithmetic logic unit (ALU) handles all internal functions, such as command setup and execution, processor timing, arithmetic and logic functions, peripheral selection and termination, and data transfer to or from memory.

The ALU contains a 16-bit parallel adder which can manipulate two 16-bit entries (four 8-bit characters) at a time (the parity bits are used for testing the validity of the bit configuration and are not manipulated by the adder). Although data is normally manipulated a byte (8 bits) at a time during command execution, the 16-bit adder permits the processor to increment 16-bit memory addresses and to perform command set-up functions two bytes at a time.

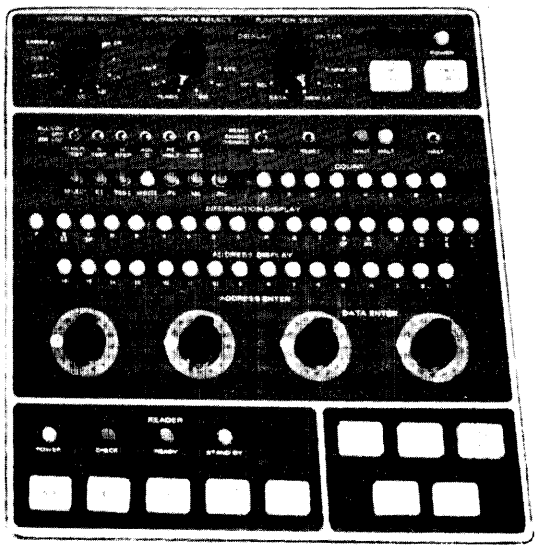
The ALU also contains live registers (hardware storage devices that are used for temporary storage of information necessary for command execution), which increase the processing efficiency of the system since their contents are readily available to the processor without requiring a memory access.

I/O Control

Simultaneous processing and I/O functions are possible with the NCR Century common-trunk concept. The ALU selects the desired peripheral, initiates the I/O function then returns to the normal processing flow while the peripheral performs offline operations such as printing, punching, reading, or writing. When the peripheral is ready to send or receive another block of data, the I/O control logic interrupts the normal processing flow and accesses the desired memory location for data transfer.

The processor uses an 8-byte control word for each peripheral. The control words keep track of the peripherals' status and determine when to terminate the I/O operation. There are 256 8-byte control words in memory: the first four are reserved for integrated peripherals, and the remaining 252 are available for common-trunk peripherals and data terminals. This large number of control words is especially valuable in realtime applications where remote terminals are being serviced by the NCR Century 621-103 Communications Multiplexer.

Operator's Console



The operator's console is the control center for system operation. It provides a means of communication between the operator and the computer, permitting the operator to display the contents of certain registers and memory locations, as well as the condition of various program flags and indicators. The console also serves as a medium through which the operator can alter memory contents and respond to software and user-program messages. An audible signal calls the operator when the processor enters either a wait or error halt state.

A detailed description of the console switches and indicators can be found in the OPERATORS INFORMATION MANUAL, HARDWARE, Consoles tab, "NCR Century 101 Console."

MEMORY

GENERAL INFORMATION

The function of the processor's memory unit is to receive information from the ALU, retain it, then release it to the processor upon request. The memory unit for the NCR Century 101 Processor is a direct-access device that uses magnetic cores for data and program storage. Each character is binarily addressable by either command or console entry. Memory, which is modular in construction, is available in the following sizes:

16,384	Characters	(16K)
24,576	Characters	(24K)
32,768	Characters	(32K)
49,152	Characters	(48K)
65,536	Characters	(64K)
98,304	Characters	(96K)
131,072	Characters	(128K)

The basic unit of information in the NCR Century 101 system is the character or byte, which is represented by nine bits of information. Eight of these bits represent a binary configuration that may be used as one 8-bit binary value, two 4-bit BCD values, or one NCR Century (ASCII) character. The ninth bit (parity bit) is used by the hardware to ensure the validity of the other eight. The hardware sets the parity bit to 1 when the other eight bits of the byte contain an even number of 1's or all 0's; this ensures an odd number of 1's in every byte and is called odd parity. The hardware checks for odd parity as each character is read from memory; if an even number of 1 bits or all 0's is detected in a byte, the hardware indicates a memory error (ME) condition.

Because the parity bit is not accessible by the program, a byte is usually referred to as an 8-bit character.

DATA REPRESENTATION

The NCR Century processor handles all data as 8-bit bytes. The eight bits are designated b8, b7, b6, b5, b4, b3, b2, b1, with b8 the most-significant and b1 the least-significant.

The information represented in an 8-bit byte may be either numeric or alphanumeric. Numeric data may be represented either as an 8-bit binary number or as two 4-bit Binary Coded Decimal (BCD) numbers. Alphanumeric data is represented as one character per byte, consisting of four zone bits and four digit bits.

Interpreting a Byte of Data

A byte of data in memory is a series of eight 1's and 0's; the processor's hardware commands determine how the bit configuration is to be interpreted. For example, the following 8-bit byte of information can be interpreted in three different ways, depending upon the command used.

ONE BYTE							
b8	b7	b6	b5	b4	b3	b2	b1
0	1	0	1	0	1	1	0

The 8-bit byte may be interpreted as two 4-bit BCD numbers, 5 and 6.

BCD POSITION VALUES							
8	4	2	1	8	4	2	1
0	1	0	1	0	1	1	0
5				6			

-or-

It may be interpreted as one 8-bit binary number with a decimal value of 86.

BINARY POSITION VALUES							
128	64	32	16	8	4	2	1
0	1	0	1	0	1	1	0
86							

-or-

It may be interpreted as an NCR Century character equivalent to the letter V (see the NCR Century code chart that follows).

NCR CENTURY CODE VALUES							
8	4	2	1	8	4	2	1
0	1	0	1	0	1	1	0
HEX 5				HEX 6			

(zone 5, digit 6 = V in the chart)

NCR Century Codes (ASCII)

The following chart illustrates the bit configurations for NCR Century characters. The chart is divided into rows and columns, with each zone-bit configuration (b8-b5) representing a row of characters and each digit-bit configuration (b4-b1) representing a column of characters. To determine the character represented by an 8-bit configuration, follow the appropriate zone-bit row across and the appropriate digit-bit column down until the two paths coincide. (The shaded row and column in the following illustration locate the NCR Century character corresponding to the bit configuration 0101 0110.)

NCR CENTURY CODE CHART

B_8 B_5 \ B_4–B_1		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	NL/LF	VT	FF	CR	SO	SI
0001	1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010	2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0011	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0110	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

The 8-bit NCR Century codes conform to the American Standard Code for Information Interchange and are, therefore, often referred to as ASCII characters.

Data Fields

Related data occupying contiguous bytes in memory may constitute a data field. There are three types of data fields: alphanumeric, decimal, and binary. The maximum size of a data field is normally 256 bytes; however, maximum field size is limited to 8 or 16 bytes for hardware MULTIPLY and DIVIDE commands, depending upon the operand (divisor limit is 8, dividend limit is 16).

- Alphanumeric Fields

  Alphanumeric fields may consist of any of the characters in the NCR Century code chart. For example, the name J. DOE would be considered an alphanumeric field.

BYTE 6		BYTE 5		BYTE 4		BYTE 3		BYTE 2		BYTE 1	
b8-b5	b4-b1	b8-b5	b4-b1	b8-b5	b4-b1	b8-b5	b4-b1	b8-b5	b4-b1	b8-b5	b4-b1
0100	1010	0010	1110	0010	0000	0100	0100	0100	1111	0100	0101
J		PERIOD		SPACE		D		O		E	

- Decimal Fields

Decimal data fields may be represented in either of two ways: (1) as unsigned and unpacked numerics (ASCII), or (2) as signed and packed numerics (BCD).

- Unsigned and Unpacked Numerics

Unsigned and unpacked decimal data fields have one decimal digit in each 8-bit byte, represented by the least-significant four bits (b4-b1); the most-significant four bits (b8-b5) are ignored by the processor when the data is manipulated and set to 0011 when the data is stored back into memory.

An unsigned unpacked field containing the decimal value of 4567 is stored in four consecutive bytes as follows.

BYTE 4		BYTE 3		BYTE 2		BYTE 1	
b8-b5	b4-b1	b8-b5	b4-b1	b8-b5	b4-b1	b8-b5	b4-b1
0011	0100	0011	0101	0011	0110	0011	0111
	4		5		6		7

- Signed and Packed Numerics

Packed decimal fields have two 4-bit decimal digits in each 8-bit byte, except in the least-significant byte of the field where the low-order bits (b4-b1) contain the sign (+ or -) of the field.

A signed packed field containing the decimal value of 04567+ is stored in three consecutive bytes as follows.

BYTE 3		BYTE 2		BYTE 1	
b8-b5	b4-b1	b8-b5	b4-b1	b8-b5	b4-b1
0000	0100	0101	0110	0111	1011
0	4	5	6	7	+

All bit configurations of the sign bits (b4-b1) except 1101 are considered to be positive; 1101 is always considered to be a negative sign. If the sign changes during an arithmetic operation, the proper ASCII sign (+ or -) is placed in the low-order bit positions of the least-significant byte.

- Binary Fields

  Binary data fields are always considered to be positive and unsigned. The maximum length of a binary field is 256 bytes. All arithmetic commands operate on binary fields one byte at a time, with a carry from the leftmost bit being added to the next byte. A carry from the leftmost byte in the field is ignored.

  The following illustration shows a 3-byte binary field containing the binary value of 69,912.

BYTE 3	BYTE 2	BYTE 1
b8-b1	b8-b1	b8-b1
00000001	00010001	00011000

69,912

ADDRESSING A FIELD IN MEMORY

Character locations in memory are numbered consecutively, starting at 0; each number indicates the address of one byte. Data fields are addressed by the leftmost byte of the field (most-significant character).

In the following example, field A, which is a 5-character field consisting of unsigned unpacked decimal data (42385), is referenced by the address 100.

ADDRESS	100	101	102	103	104
CONTENTS	4	2	3	8	5

Field A

If the same data is divided into two fields (A and B), the address of field A is 100, and the address of field B is equal to the address of the most significant character in that field.

ADDRESS	100	101	102	103	104
CONTENTS	4	2	3	8	5

Field A is located at address 100. Field B is located at address 103.

The addresses in the preceding examples were given in decimal form for simplicity. Actual memory addresses, however, are in binary form, consisting of 16 bits (two bytes) per address. Each bit has a binary value as indicated in the following illustration.

BINARY VALUES FOR MEMORY ADDRESSING															
MOST-SIGNIFICANT BYTE								LEAST-SIGNIFICANT BYTE							
b8	b7	b6	b5	b4	b3	b2	b1	b8	b7	b6	b5	b4	b3	b2	b1
32,768	16,384	8,192	4,096	2,048	1,024	512	256	128	64	32	16	8	4	2	1

To address decimal location 4536, the following binary configuration must be setup:

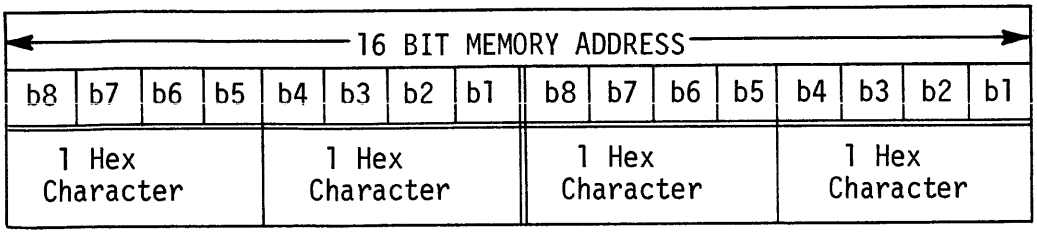
MOST-SIGNIFICANT BYTE								LEAST-SIGNIFICANT BYTE							
b8	b7	b6	b5	b4	b3	b2	b1	b8	b7	b6	b5	b4	b3	b2	b1
0	0	0	1	0	0	0	1	1	0	1	1	1	0	0	0

To address decimal location 65,535, all bits must be set to 1 (memory is addressed from 0 to 65,535, providing 65,536 locations). Attempting to access a memory location equal to or greater than the physical memory size results in a program error (PE), except on systems with 64K memory (no address larger than 65,535 can be obtained because the processor recognizes only 2-byte memory addresses). A PE is created whenever a memory address is incremented beyond the physical memory size.

Memory may be addressed either by manual entry through the operator's console or by hardware command.

Addressing Memory By Console Entry

The NCR Century 101 operator's console contains four ADDRESS-ENTER dials that may be used to address any byte in memory. These dials employ the hexadecimal numbering system in which each dial represents four bits of binary information (one hexadecimal character). The four ADDRESS-ENTER dials collectively represent a 16-bit memory address.



The following conversion table illustrates the relationship between decimal, binary, and hexadecimal characters.

CONVERSION TABLE		
DECIMAL	BINARY	HEX
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

To enter a memory address through the operator's console, the user must first convert it from either decimal or binary values to four equivalent hexadecimal characters.

- Converting Binary/Hexadecimal

  To convert from binary to hexadecimal, simply separate the binary address into four 4-bit groups and assign the appropriate hexadecimal digit to each group:

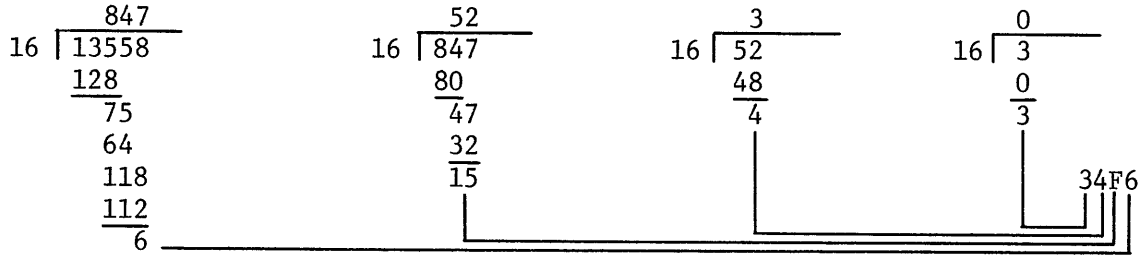
Binary Value		Conversion					Hex Value
0011010111101010	=	0011	0101	1110	1010	=	35EA
		3	5	E	A		

  To convert from hex to binary, reverse the procedure:

Hex Value		Conversion					Binary Value
2F9A	=	2	F	9	A	=	0010111110011010
		0010	1111	1001	1010		

- Converting Decimal/Hexadecimal

To convert a decimal value to a hexadecimal equivalent, the user may systematically divide by 16 until the quotient is 0, using the remainder after each division as the hexadecimal value. The quotient from each division is used as the dividend in the next division. The decimal value of 13,558 is converted to hex 34F6 as follows.



To convert a 4-character hexadecimal value to a decimal equivalent, the user may expand the value of each hex character to the base of 16 as shown in the following example:

$$34F6 = (3 \times 16^3) + (4 \times 16^2) + (15 \times 16^1) + (6 \times 16^0)$$
$$= (3 \times 4096) + (4 \times 256) + (15 \times 16) + (6 \times 1)$$
$$= 12{,}288 + 1024 + 240 + 6$$
$$= 13{,}558$$

For simplicity, the user may use the following table to convert between decimal and hexadecimal.

HEXADECIMAL/DECIMAL CONVERSION

HEX EQUIVALENT	DECIMAL VALUES			
	COLUMN 4	COLUMN 3	COLUMN 2	COLUMN 1
0	0	0	0	0
1	4,096	256	16	1
2	8,192	512	32	2
3	12,288	768	48	3
4	16,384	1,024	64	4
5	20,480	1,280	80	5
6	24,576	1,536	96	6
7	28,672	1,792	112	7
8	32,768	2,048	128	8
9	36,864	2,304	144	9
A	40,960	2,560	160	10
B	45,056	2,816	176	11
C	49,152	3,072	192	12
D	53,248	3,328	208	13
E	57,344	3,584	224	14
F	61,440	3,840	240	15

To convert from decimal to hexadecimal using the chart, the user must perform a series of subtractions. Consider the following example to convert the decimal value of 13,558 to a hexadecimal equivalent of 34F6.

1. Subtract from the desired decimal value the largest applicable amount in column 4. Note the hex equivalent (3).

13,558	Decimal value
-12,288	Amount in col. 4
1,270	Remainder

2. Subtract from the remainder the largest applicable amount in column 3. Note the hex equivalent (4).

1,270	Remainder
-1,024	Amount in col. 3
246	Remainder

3. Repeat step 2, using amounts from columns 2 and 1. Note the hex equivalent for each column (F for column 2 and 6 for column 1).

246	Remainder
-240	Amount in col. 2
6	Remainder
6	Remainder
-6	Amount in col. 1
0	

4. The hex equivalent of 13,558 is 34F6.

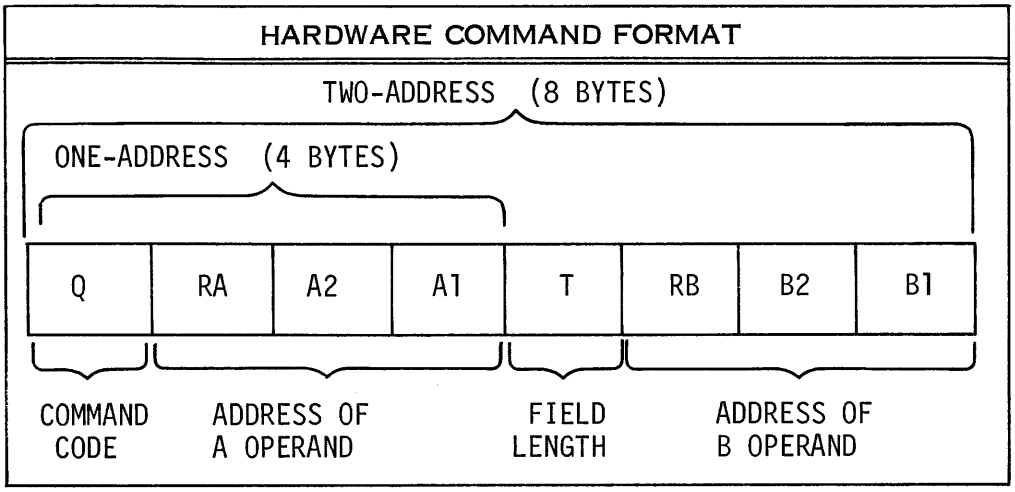
To convert from hexadecimal to decimal using the chart, the user must select the appropriate decimal value from each column, then add the values together. For example, hex 3FOB is converted to decimal 16,139 as follows:

Column 4 equivalent of hex 3 =	12,288
Column 3 equivalent of hex F =	3,840
Column 2 equivalent of hex 0 =	0
Column 1 equivalent of hex B =	11
	16,139

Addressing Memory By Hardware Command

During program execution, memory locations are accessed by hardware commands (object program commands). Although hardware commands deal basically with two operands (A and B), they may be coded either as one-address commands or as two-address commands. Both types of commands are fully equivalent in action; the only difference is that when a second operand is needed with a one-address command, it is obtained from the last executed two-address command. For example, after executing a two-address command, the processor leaves the address of the B operand in a hardware register where it is available for a subsequent one-address command. When the one-address command is executed, it uses the implied B address from the two-address command and again leaves an implied B address for the next command (the implied B address left by the one-address command may be the same as the one left by the two-address command, or it may be different, depending upon the characteristics of the one-address command). Each two-address command also specifies the length of the fields being accessed.

The following illustration shows both command formats.



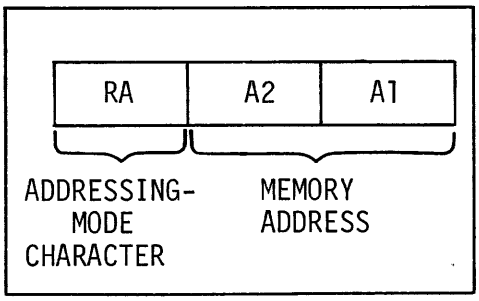
- Command Code (Q)

  The command code (Q) is a 1-byte field which specifies the command to be executed and its format (one-address or two-address). The most-significant bit (b8) determines the format (0 designates a two-address command, and 1 designates a one-address command). The binary configuration of the remaining seven bits (b7-b1) designates the type of command to be executed (add, subtract, etc.).

- A-Operand Field

  The A-operand field consists of an addressing-mode character (RA) and a 2-byte memory address (A2A1).

  RA serves two functions; it contains the address of the index register that may be used, and it designates one of four addressing modes (the two least-significant bits -- b2 and b1 -- specify the addressing mode, and the remaining bits determine the address of the index register).



  The 2-byte memory address (A2A1) may be used by itself to designate the beginning address of a memory field, or it may be used in conjunction with the contents of a 2-byte index register. If A2A1 represents only a partial memory address, RA contains the address of the index register which is to be used in creating the effective address.

- Field Length (T)

  The T field is an 8-bit binary character, ranging in value from 0 to 255, with 0 usually considered equal to 256. The function of this field varies with the type of command; however, it normally indicates the length of the A operand, the length of the B operand, or both. (To determine the function of the T field in each command, refer to the NCR Century 101 Hardware Command and Command Timing Publication in this manual.)

  When the T field indicates the length of both the A and B operands, the NCR Century compilers generate the coding necessary to ensure that A and B fields are the same length before command execution.

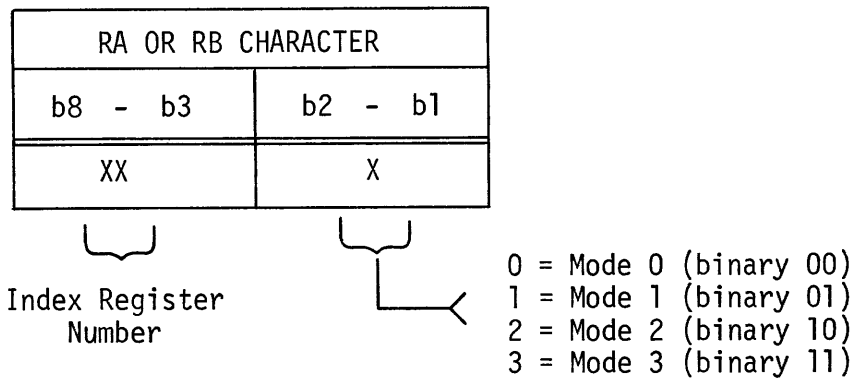
- B-Operand Field

  The B-operand field, like the A-operand field, consists of an addressing-mode character (RB) and a 2-byte memory address (B2B1). The function of each character in the B operand is the same as the equivalent character in the A operand.

Special Addressing Modes

The addressing-mode characters of the A and B operands (RA and RB) determine the manner in which the effective memory addresses are to be calculated. Because each operand has its own addressing-mode character, it is possible for the A operand to use one mode of addressing and the B operand to use another.

Four addressing modes are available for the A and B operands: Mode 0 (direct addressing), Mode 1 (indirect addressing), Mode 2 (indirect addressing), and Mode 3 (incremental addressing). For simplicity in discussing the four modes, decimal notations are used and RA or RB is considered as a 2-part character, the first part being an index register designation and the second part being a mode designation.

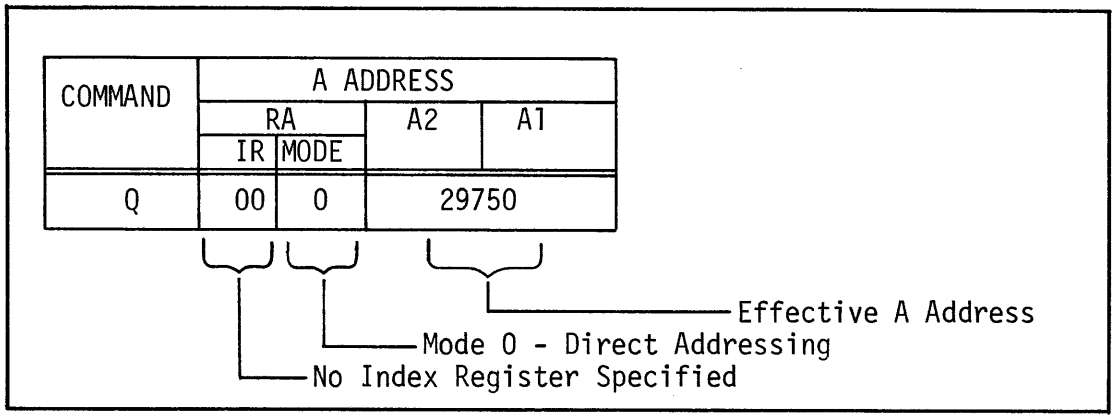


To further simplify the discussion, only the A operand is considered; addressing for the B operand is identical, except that the RB addressing-mode character is used with B2B1.

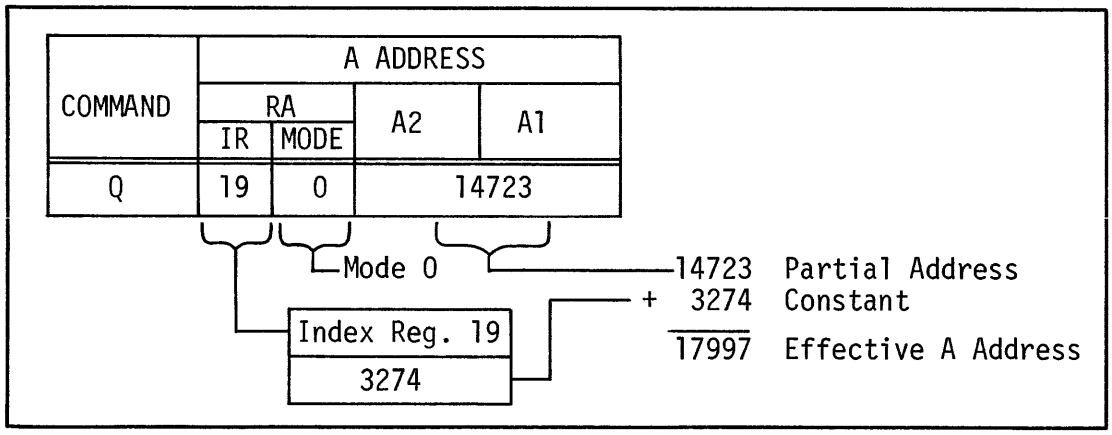
- Mode 0 Addressing

Mode 0 provides the programmer with direct-addressing capabilities; that is, he may specify the beginning address of the A field in the A2A1 portion of the command, or he may specify the address of an index register which may be used with the A2A1 portion of the command to create an effective A-field address.

If no index register is specified (b8-b3 of RA = 0), the A2A1 portion of the command contains the effective A address.



If an index register is specified (b8-b3 of RA ≠ 0), the bit configuration of RA selects the index register in memory that contains a constant which may be used to point to the beginning of a memory field. The binary value of A2A1 is then added to the index register constant to obtain the effective A address.

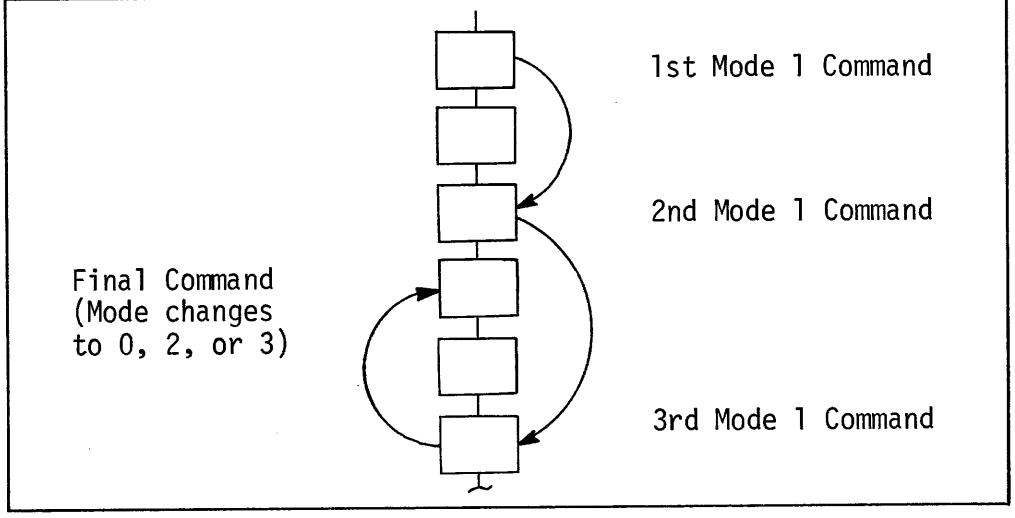


- Mode 1 Addressing

Mode 1 permits the user to indirectly access an area of memory by using the addressing mode of another command or string of commands. (The area referenced by Mode 1 need not be another command, but it must be a 4-byte area in memory, at an address that is evenly divisible by 4; in addition, the contents of this area must assume a command format with a 3-byte operand -- RA, A2, A1, or RB, B2, B1 -- in the low-order three bytes.) If the addressing mode of another command is used, Mode 1 addressing for the A operand references the first four bytes of the command, while Mode 1 addressing for the B operand references the last four bytes.

Reference may be made to five successive commands, providing each of the commands, in turn, specify Mode 1 addressing; if the mode of addressing changes in any of the intermediate commands, the rules governing that particular mode are followed until addressing setup is complete. Attempting to reference beyond five consecutive commands in Mode 1 results in a program error (PE).

EXAMPLE OF 4—LEVEL INDIRECT ADDRESSING



For simplicity, the following explanation considers only Mode 1 addressing for the A operand; Mode 1 addressing for the B operand is identical in theory, the only difference being that T, RB, B2 and B1 values are used instead of Q, RA, A2, and A1.

Each command in the flow may or may not specify an index register, depending upon the binary value of RA. If b8-b3 = 0, no index register is specified and A2A1 becomes the effective indirect address which points to the next command in the sequence. If b8-b3 ≠ 0, A2A1 is added to the value of the specified index register to form an effective indirect address which points to the next command in the sequence.

Mode 1 W/O Index Register

COM	A ADDRESS			
	RA		A2	A1
	IR	MODE		
Q	00	1	6384	

1st Effective Indirect Address

Mode 1 With Index Register

COM	A ADDRESS			
	RA		A2	A1
	IR	MODE		
Q	19	1	1024	

1024 Partial A Address

+ 5360 Constant

6384

IR19
5360

1st Effective Indirect Address

Each command in the sequence may initiate another stage in the setup procedure (by requesting Mode 1 addressing), or it may terminate the sequence by requesting Mode 0, 2, or 3 addressing.

FINAL COMMAND IN MODE 1 SEQUENCE

Indirect address established by the previous Mode 1 command.

COMMAND	RA		A2	A1
	IR	MODE		
Q	XX	X	10248	

- If RA specifies no index register and the addressing mode changes to 0, the effective A address is 10248.
- If RA specifies an index register and the mode changes to 0, the effective A address is equal to the constant stored in the specified index register, plus 10248.
- If RA specifies Mode 2 or 3, that mode of addressing is performed, using the values in this command to obtain an effective A address.

- Mode 2 Addressing

Mode 2 permits the user to indirectly access the desired A field by addressing a 3-character constant stored in another area of memory; the two least-significant characters of the constant are used to designate the effective address of the desired A field (the most-significant character is not used). The address of the constant (called the effective indirect address) is equal to the content of the specified index register, if any, plus A2A1; this address need not be evenly divisible by 4.

Mode 2 W/O Index Register

COM	A ADDRESS			
	RA		A2	A1
	IR	MODE		
Q	0	2	8507	

Effective Indirect Address

Mode 2 With Index Register

COM	A ADDRESS			
	RA		A2	A1
	IR	MODE		
Q	60	2	2048	

2048 Partial A Address

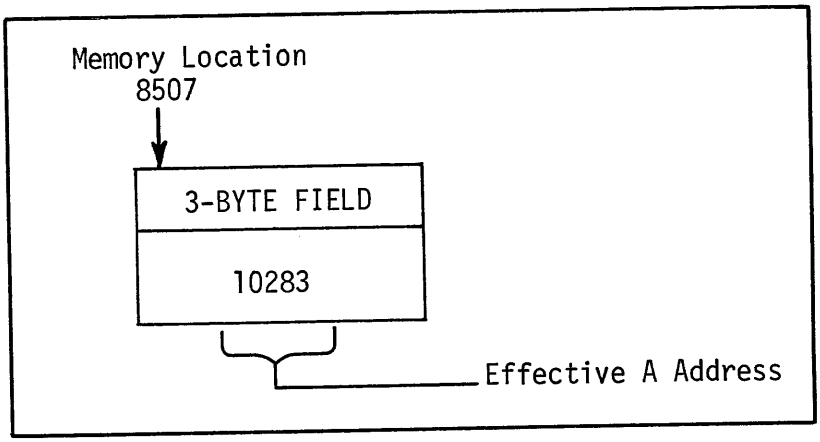
+ 6459 Constant

IR60
6459

8507

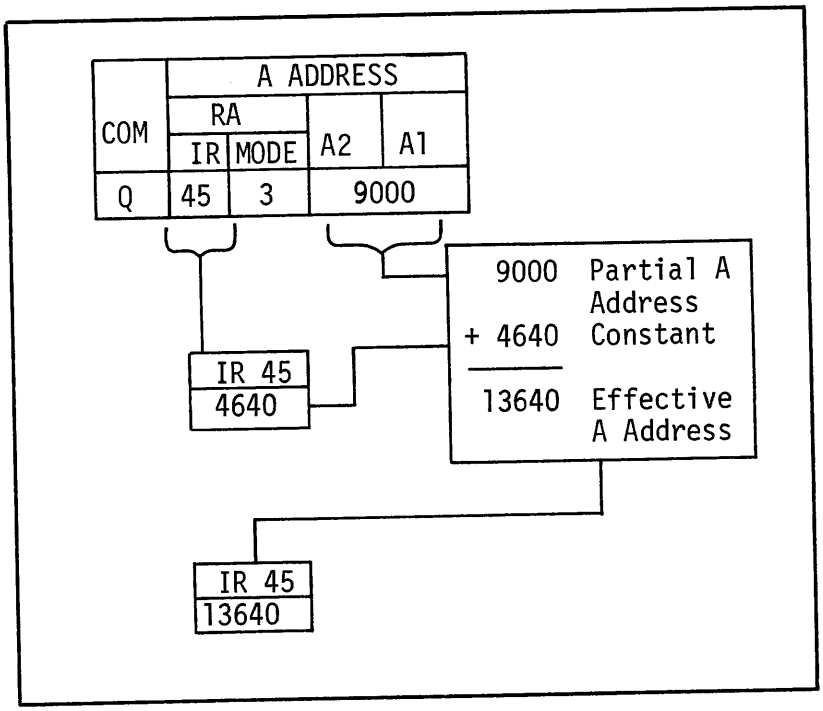
Effective Indirect Address

In either case, the effective indirect address points to the 3-byte field in memory that contains the constant to be used as an effective A address.

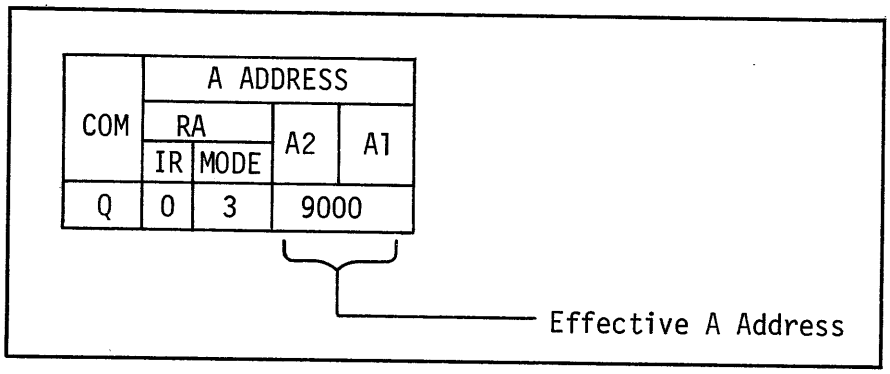


- Mode 3 Addressing

Mode 3 permits the user to step through any area of memory by byte or by field, such as stepping through a table. In this mode, the first effective A address is equal to the contents of the specified index register, plus the value of A2A1; however, after command setup is complete, the content of the index register is incremented by the value of A2A1. The next time this command is setup, the new constant is used to compute the next effective A address.



If no index register is specified with Mode 3 addressing, the value of A2A1 immediately becomes the effective A address, and the command is executed as though Mode 0 addressing was requested.



AREAS RESERVED FOR HARDWARE FUNCTIONS

For compatibility with other members of the NCR Century family, the first 3584 bytes of memory are normally reserved for system use. The following memory map identifies (by decimal and hexadecimal notations) the first 3,071 memory locations and specifies their use wherever applicable. Because the software requires these areas to be addressed with 0 modulo 4 addressing, each line represents a 4-byte field with the most-significant byte (byte 4) as the leftmost character. The use of each area is explained following the memory map.

NCR CENTURY 101 RESERVED MEMORY AREAS

	FUNCTION	DECIMAL LOCATION	BYTE 4 b8→b1	BYTE 3 b8→b1	BYTE 2 b8→b1	BYTE 1 b8→b1	HEX LOCATION
		0000 - 0003					0000 - 0003
RESERVED FOR SOFTWARE CONTROL	Index Reg. Words (1-4) Reserved for upward software compatibility from smaller NCR Century Systems.	0004 - 0007	Reserved		Reserved	Reserved	0004 - 0007
					Software Use	(Index Reg. 1)	
		0008 - 0011	Reserved		Reserved	Reserved	0008 - 000B
					Software Use	(Index Reg. 2)	
		0012 - 0015	Reserved		Reserved	Reserved	000C - 000F
					Software Use	(Index Reg. 3)	
		0016 - 0019	Reserved		Reserved	Reserved	0010 - 0013
					Software Use	(Index Reg. 4)	
	Index Reg. Words (5,6,7) Used to store error status on ME, PE, or command code trapping.	0020 - 0023	Q: Command Code		A2 / Effective A	A1 (Index Reg. 5)	0014 - 0017
		0024 - 0027	T: Field Length		B2 / Effective B	B1 (Index Reg. 6)	0018 - 001B
		0028 - 0031	OF RI G E L: Flags		CR2: Control Register	CR1 (Index Reg. 7)	001C - 001F
	Index Reg. Words (8-9) Special Reg.	0032 - 0035	RC: Repeat Counter		L2 / Return Link	L1 (Index Reg. 8)	0020 - 0023
		0036 - 0039	EC: Error Code		N2 / Next Address	N1 (Index Reg. 9)	0024 - 0027
CONTROL	Index Reg. Words (10,11,12) Used for program interrupt.	0040 - 0043	Q: Command Code		A2 / Effective A	A1 (Index Reg. 10)	0028 - 002B
		0044 - 0047	T: Field Length		B2 / Effective B	B1 (Index Reg. 11)	002C - 002F
		0048 - 0051	OF RI G E L: Flags		CR2: Control Register	CR1 (Index Reg. 12)	0030 - 0033
WORK REG	Index Reg. Words (13 - 63)	0052 - 0055			Work Register	(Index Reg. 13)	0034 - 0037
		0056 - 0059			Work Register	(Index Reg. 14)	0038 - 003B
		0248 - 0251			Work Register	(Index Reg. 62)	00F8 - 00 FB
		0252 - 0255			Work Register	(Index Reg. 63)	00FC - 00FF
ERROR CONTROL	ME CONTROL AREA	0256 - 0259			ME2 / ME1: Address of Memory Error Routine		0100 - 0103
	PE/ICC CONTROL AREA	0260 - 0263			PE2 / PE1: Address of Program Error Routine		0104 - 0107
	Reserved for software use	0264 - 0267					0108 - 010B
		0268 - 0271					010C - 010F
IP	INTERRUPT CONTROL	0272 - 0275			I2 / I1: Address to receive control when interrupt trap occurs.		0110 - 0113
	Reserved for	0276 - 0279					0114 - 0117
	Software Use	0284 - 0287					011C - 011F
CONTROL	Divisor Accumulator Used by the hardware DIVIDE command	0288 - 0291					0120 - 0123
		0292 - 0295					0124 - 0127
		0296 - 0299					0128 - 012B
		0300 - 0303					012C - 012F
	Dividend Accumulator Used by the hardware DIVIDE command	0304 - 0307					0130 - 0133
		0308 - 0311					0134 - 0137
		0312 - 0315					0138 - 013B
		0316 - 0319					013C - 013F
	Memory Accumulator Used by the hardware MULTIPLY and DIVIDE commands to contain the result.	0320 - 0323					0140 - 0143
		0324 - 0327					0144 - 0147
		0328 - 0331					0148 - 014B
		0332 - 0335					014C - 014F
	Reserved for	0336 - 0339					0150 - 0153
	Software Use	1020 - 1023					03FC - 03FF
PERIPHERAL CONTROL WORDS	Control Word 0 Used for the COT.	1024 - 1027	S: Status Code		N2 / N1: Next Address (NA)		0400 - 0403
		1028 - 1031	T2 / T1: Terminating Address (TA)				0404 - 0407
	Control Word 1 Used for the operator's console	1032 - 1035	S: Status Code		N2 / N1: Next Address (NA)		0408 - 040B
		1036 - 1039	T2 / T1: Terminating Address (TA)				040C - 040F
	Control Word 2 Used for the integrated printer	1040 - 1043	S: Status Code	H: Row Char. Image	B2 / B1: Beginning Address (BA)		0410 - 0413
		1044 - 1047	C: Char. Count	I/N: Counter			0414 - 0417
	Control Word 3 Used for the I/O writer	1048 - 1051	S: Status Code		N2 / N1: Next Address (NA)		0418 - 041B
		1052 - 1055	T2 / T1: Terminating Address (TA)				041C - 041F
	Control Words (4-254)						
	Control Word 255	3064 - 3067	S: Status Code		N2 / N1: Next Address (NA)		0BF8 - 0BFB
		3068 - 3071	T2 / T1: Terminating Address (TA)				0BFC - 0BFF

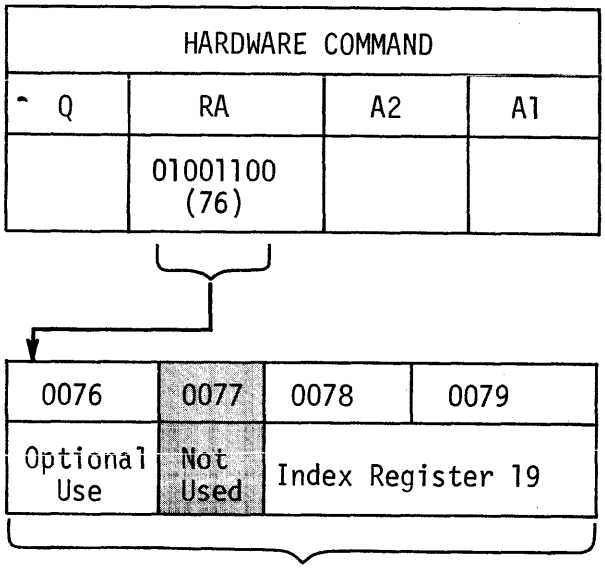
Index Register Words

The NCR Century 101 memory contains 63 index registers, designated IR1 through IR63. Each index register is located in the two rightmost bytes (bytes 2 and 1) of a 4-byte index register word. The leftmost byte (byte 4) of the index register word is normally used to contain related flags, codes, indicators, etc.; byte 3 is used only on larger NCR Century Systems to indicate memory addresses beyond 65,536 and should be set to zero for upward compatibility.

INDEX REGISTER WORD			
b8 - b1 (Byte 4)	b8 - b1 (Byte 3)	b8 - b1 (Byte 2)	b8 - b1 (Byte 1)
Optional Use	Not Used	Index Register	

Index register words are located consecutively in memory locations 0004 through 00FF. IR1 through IR12, IR21 through IR39, IR62 and IR63 are reserved for system software use. In addition, IR13, IR14, and IR15 should be reserved for upware compatibility with larger NCR Century systems. The remaining index registers (IR16 through IR20 and IR40 through IR61) can be used by the programmer, providing their assignment does not conflict with special software assignments. If the user allows the compiler to assign index registers, the assignment is done on a next available basis; however, if he assigns his own index registers and happens to pick one that has been assigned to a particular software routine, a compiler error is generated.

The RA or RB field of each hardware command addresses the leftmost byte of an index register word (bits 1 and 2 are used only to determine the addressing mode; bits 8-3 indicate the address of the index register word). An offset of 2 (binary) is added by the arithmetic logic unit to obtain the address of the most-significant byte of the index register (byte 2 of the index register word).



Index Register Word 19

ME, PE/ICC and Program Interrupt Control Areas

The memory error (ME), program error/illegal command code (PE/ICC), and program interrupt control areas (each four bytes in length) contain addresses to which control is given whenever an error is detected or when program interrupt is desired.

The ME control area (hex 0100-0103) contains the address of the first command in a software routine to which control is to be given in the event of a memory error or ROM (read-only-memory) error. (Memory errors and ROM errors are explained in the arithmetic logic section of this publication, under the headings of Memory Error Indicator and Read-Only-Memory Indicator.) Either type of error causes the processor to enter a hardware trapping flow. During the trapping flow, the state of the processor is stored in index register words 5, 6, and 7, where it is available for recovery purposes. The type of error is indicated by setting the EC (error code) flag (hex 0024) to hex 00 for a memory error or to hex 04 for a ROM error. The beginning address of the software recovery routine is then transferred from the ME control area to the sequence control register (CR) and control is given to that routine (providing that no additional errors are detected during the trapping flow).

The PE/ICC control area (hex 0104-0107) contains the address of the first command in a software routine to which control is to be given in the event of a program error or an illegal command code (program errors and illegal command code conditions are explained in the arithmetic logic section of this publication, under the headings of Program Error Indicator and Illegal Command Code Indicator). When either condition is encountered, the processor enters a hardware trapping flow. During the trapping flow, the state of the processor is stored in index register words 5, 6, and 7, where it is available for recovery purposes. The type of error is indicated by setting the EC (error code) flag (hex 0024) to a unique bit configuration: PE = hex 01, ICC = hex 02, and PE/ICC = hex 03. Finally, the beginning address of the software routine is transferred from the PE/ICC control area to the sequence control register (CR) and control is given to that routine (providing that no additional errors are detected during the trapping flow).

The interrupt control area (hex 0110-0113) contains the address of the first command in a software routine to which control is to be given in the event of a program interrupt to process the termination of an I/O operation. When the processor detects the termination of an I/O operation with the interrupt permit (IP) flag ON, it enters a hardware trapping flow, which stores the status of the processor in index register words 10, 11, and 12, then transfers the beginning address of the software interrupt routine from the interrupt control area to the sequence control register (CR). Finally, the processor gives control to the software routine.

Divisor and Dividend Accumulators

The divisor and dividend accumulators (hex 0120-013F) are used during the execution of the hardware DIVIDE command. These accumulators contain different information at various stages of command execution, depending upon the length of the divisor and dividend.

Memory Accumulator

The memory accumulator (hex 0140-014F) is a 16-byte area in memory that may be used to contain either the product of a hardware MULTIPLY command, or the quotient and remainder of a hardware DIVIDE command.

Decimal multiplication is performed on packed signed fields. The product of the multiplication is then right-justified and placed in the memory accumulator; all characters to the left of the product are unchanged. The length of the product is equal to the sum of the lengths of the A and B operands.

Decimal division is performed on packed signed fields. The quotient and remainder are both placed in the memory accumulator, with the quotient field being 8 bytes (15 digits and sign) or less and right-justified in the accumulator; the remainder field is left-justified in the accumulator with a maximum length of 8 bytes (15 digits and sign).

Control Words

There are 256 8-byte control words which may be used by the I/O control to (1) address information, (2) determine when to terminate an operation, and (3) store status characters. These control words begin at memory location 1024 (hex 0400).

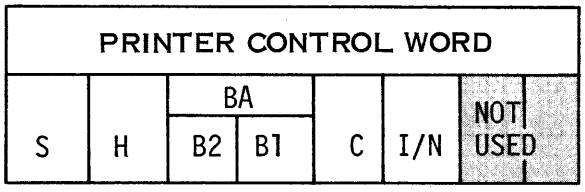
At installation time, each peripheral is assigned a unique response number, which is used by the I/O control to locate the corresponding control word. For example, response number 0 selects address 1024 (control word 0), response number 1 selects address 1032 (control word 1), etc.

There are two control word formats: one for the integrated printer and another for all other peripherals.

- Integrated Printer Control Word

  The integrated printer control word (CW2) begins at memory location hex 0410 and contains the following information: start of the print area in memory, number of characters to print, number of lines to advance, and status of the operation.

  The following illustration describes the format of the integrated printer control word:



  S - Storage location for a status character which indicates the success or failure of the I/O operation. (See the PRINTERS tab, "640-102 Printer," or "640-300/301 Printer" in this manual for more information on status characters.)

H - Used by the hardware to store the row character image.

BA - Beginning memory address of the print area.

C - Not used for the NCR 640-300 Integrated Printer, nor for the NCR 640-102 Integrated Printer equipped with a single-numeric printline. For the 640-102 Integrated Printer equipped with a double-numeric printline, this character designates the number of graphic positions on the typeline over which printing may occur. If this character is 13 or less, only the numeric set of characters can be printed; if it is greater than 13, a full set of alphanumeric characters is assumed and termination is controlled by the printer (see the PRINTERS tab, "640-300/301 Printer" in this manual for a description of the software character sets).

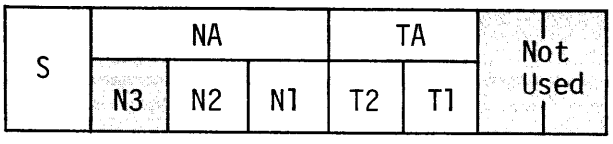
I - For the 640-102 Integrated Printer, this character contains the number which is added to BA during the printing of a line in order to select the different columns on the typeline (see the PRINTERS tab, "640-102 Printer" for more information on selecting print columns). For the 640-300 Integrated Printer, this character contains a number which is decremented by one each time a new character comes into printing position; when it has been decremented to zero (indicating that the typeline has made a complete revolution), the printer control terminates the print function. Printing may also be terminated if all print hammers are fired before the I character is decremented to zero.

N - A 1-byte binary counter which may be set from 0 to 255 to indicate the number of lines to advance (0 = 256). Line advance is complete when N equals 0, or when the page sentinel is detected.

- Other Peripheral Control Words

All other peripherals use the following control word format to (1) address information, (2) determine when to terminate an operation, and (3) store the status character. Control words 0, 1, and 3 are permanently assigned to the COT (punched card or punched tape reader), the console switches, and the I/O Writer or thermal I/O Writer respectively; all other control words (with the exception of control word 2 which is assigned to the integrated printer) may be assigned as needed.

The following illustration describes the format of peripheral control words other than the printer control word:



S - Storage location for the status character (S3 or S4) which indicates the result of the operation. (See the I/O control section in this publication for further explanation of status characters.)

NA - Address of the next character to be accessed by the I/O control. Only N2 and N1 are used for the NCR Century 101 System; however, for compatibility with larger NCR Century systems, N3 should be set to 0.

TA - Terminating address representing the last address to be accessed, plus one. After each character is accessed by the I/O control, the NA address (N2N1) is incremented by one and compared to the TA address. When NA = TA, the processor terminates the operation. The NA/TA arrangement allows a maximum record size of 65,536 bytes.

ARITHMETIC LOGIC UNIT

GENERAL INFORMATION

The arithmetic logic unit (1) controls command setup and execution, (2) controls memory addressing, (3) regulates the transfer of data to and from memory, (4) controls processor timing, and (5) performs all arithmetic and logic functions. In effect, the arithmetic logic unit (ALU) handles all of the processor's internal functions.

For simplicity, the ALU is discussed in five parts: the control section, hardware registers, hardware flags and indicators, the adder, and functional operation.

ALU CONTROL SECTION

The ALU control section contains a special read-only-memory (not to be confused with the processor's internal memory unit) that consists of integrated logic circuitry and certain preset values. These values, which are established at the time of manufacture, can in no way be changed during the execution of software or user programs. The processor may, however, input other values to specific circuits to achieve predetermined outputs (as in multiply and divide functions), or it may address specific circuits to establish processor timing and sequence control. This type of memory is called a read-only-memory (ROM) because its contents can only be accessed; they cannot be changed.

The ALU control section performs three distinct functions: it provides (1) processor timing, (2) normal control logic, and (3) miscellaneous decision logic.

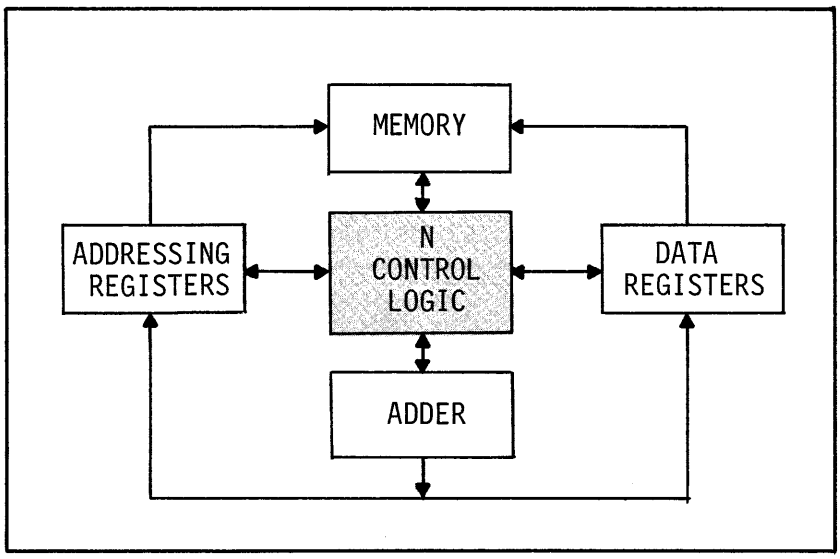
Processor Timing

The NCR Century 101 Processor is synchronized to the memory cycle, creating a processor cycle (sometimes called a processor count) equal to 1.2 microseconds in length. The processor and memory both cycle continuously whenever the processor power is ON, providing that the processor is not in a halt state. Information in memory is accessed only when the control logic addresses it.

For processing efficiency, each processor cycle is divided into two adder cycles (PA and PB), making it possible for the processor to input data to the adder twice during each cycle. For example, the processor may input data to the adder from any of the hardware registers during the PA adder cycle and information from memory during the PB adder cycle, or it may input data from the hardware registers during both adder cycles.

Normal Control Logic

The normal (N) control logic regulates data transfer among the hardware data registers, to and from memory, through the adder, and to the addressing registers.



The logic necessary to perform all of these functions is divided into small groups called N flows, with each flow performing a specific function relative to the setup and execution of hardware commands. For example, the first N flow (N-01) accesses memory to obtain the first four bytes of the command, places the Q code into a hardware register where it can be decoded, obtains the address of the index register (if any), determines the mode of addressing for the A operand, and obtains the partial A address (A2A1).

The next N flow to be accessed is determined by the miscellaneous decision logic; is not necessarily the next flow in numerical sequence. The processor advances from flow to flow until the command has been completely setup and executed; it then performs Between Commands Testing (BCT) to determine whether the command was successfully executed.

If an error condition is detected during the BCT check, the processor may either enter an appropriate error trapping flow (to preserve the status of the processor and transfer control to a software recovery routine), or it may enter a halt state and terminate all processing. If no error condition is detected, the processor normally returns to flow N-01 to begin setup for the next command.

Miscellaneous Decision Logic

The miscellaneous decision logic guides the processor through the necessary N-flows to accomplish the desired function. It also controls special logic for particular commands.

HARDWARE REGISTERS

The NCR Century 101 uses hardware registers as temporary storage devices, which may be accessed both independently and simultaneously with memory. Because hardware registers can be accessed independently of memory, they are sometimes called live-registers.

The following table lists the hardware registers that are most commonly referred to in this publication and indicates the normal use for each one; special function registers are not included.

COMMONLY USED HARDWARE REGISTERS		
TERM	NAME	FUNCTION
CR	Control Register	Normally contains the address of the next command to be executed.
LA	A Address Register	Used to build and contain the effective address for the A operand.
LB	B Address Register	Used to build and contain the effective address for the B operand.
LC	Miscellaneous Register	Used during command setup and execution as a temporary storage device.
L	Memory Address Register	Used to address memory (all addressing is accomplished through this register).
MA	Adder Input Register	16-bit register used to circulate data through the adder.
MAP	Adder Input Register	8-bit register used to circulate data through the adder.
MB	Memory Input Register	8- or 16-bit input to memory.
T	T Register	Used to contain the T value of the command.
TC	Tally Register	Used to create tally counts.
Q	Command Code Register	Used to contain the Q code of the command being executed.

FLAGS AND INDICATORS

Embodied in the arithmetic logic unit is a set of hardware flags and indicators which can be used by the processor to denote a condition or the result of an operation. These hardware flags and indicators (not to be confused with memory flags and indicators) are 2-state electrical circuits which may be either ON or OFF, with ON meaning that a certain condition or result has occurred. Hardware flags and indicators, like hardware registers, are readily available to the processor and may be accessed either independently or simultaneously with memory.

Comparison Flags (L, E, and G)

The three comparison flags (less, equal, and greater) and their associated console lights indicate the result of a comparison or decode operation. Testing of these flags by the various BRANCH commands does not change their status; therefore, once a flag is set ON, it remains in that state until one of the other flags is set ON or until the RESTORE command is executed.

The RESTORE command reestablishes the status of these flags following an error condition or program interrupt. This command is normally used to set the hardware comparison flags according to the status of the memory comparison flags (index register word 7 or 12), but it can also be used to set the comparison flags according to any other status word in the user's program.

The following table lists all hardware commands that influence the hardware comparison flags and indicates the effect that each command has on specific flags.

COMMAND CONTROL OF L, E, AND G FLAGS

COMMAND	CONTROL
RESTORE	Sets the flags ON or OFF according to the status of the memory L, E, and G flags or any user defined status word.
COMPARE BINARY and COMPARE SIGNED	Sets the flags ON or OFF according to the result of the comparison.
DECODE ALL	Sets the G flag ON and the other two flags OFF.
DECODE TO DELIMITER	Sets the E flag ON when a delimiter character (bit 8 ON) is encountered in the A field. Sets the G flag ON if no delimiter character is encountered. (The L flag is always set OFF.)

Overflow Flag (OF)

The overflow flag and its associated console light are set ON whenever the result of an ADD, SUBTRACT, ADD UNSIGNED, SUBTRACT UNSIGNED, or DIVIDE operation exceeds the specified field size. This flag is used by the processor to detect a negative result, to detect an illegal BCD result, to branch to an overflow routine, etc.

The overflow flag is set OFF by the BRANCH OVERFLOW command and any of the trapping flows (ME, PE, ICC, and Program Interrupt). In addition, the overflow flag can be set ON or OFF by the RESTORE command, depending upon the status of the memory overflow flag in index register word 7 or 12 or any other status word in the user's program.

Repeat Indicator (RI)

The repeat indicator and its associated console light are set ON by the REPEAT command. They are used by the processor to indicate whether the next command in sequence should be repeated. When this indicator is ON, the processor repeats setup and execution of the next command in sequence.

Repeated setup and execute functions continue until the repeat counter (memory location hex 20) is decremented to zero or until one of the following conditions is encountered:

- Any one of the BRANCH commands is executed.
- The WAIT command is executed.
- The INOUT command is executed.
- The SET IP ON command is executed.
- The SET IP OFF command is executed.
- The JUMP command is executed.
- Any one of the trapping flows (ME, PE, ICC, or Program Interrupt) is initiated.
- The COMPARE BINARY command is executed with A equal to or greater than B.
- The SIGNED COMPARE command is executed with A equal to or greater than B.
- The TEST EQUAL command is executed with A equal to B.
- The TEST UNEQUAL command is executed with A not equal to B.
- The TEST BIT command is executed with either the 1 bits in T equal to the 1 bits in B, or no 1 bits in T.

The repeat indicator may also be set OFF by the console LOAD switch, or set according to the status of the memory repeat indicator (index register word 7 or 12) by the RESTORE command (the RESTORE command may also set the repeat indicator according to the content of a status word set up by the user's program).

Interrupt Permit Indicator (IP)

The interrupt permit indicator and its associated console light are set ON by the SET IP ON command. They are used by the processor to indicate whether it is permissible to interrupt the program flow for peripheral I/O termination.

At the completion of each command execution (except SET IP ON), the processor checks the interrupt permit indicator (IP) and the interrupt indicator (II). If both indicators are ON, the processor enters the interrupt trapping flow; otherwise, processing continues with the next command in sequence.

The interrupt permit indicator (IP) may be set OFF by any of the following conditions:

- Execution of the SET IP OFF command.
- Initiation of the interrupt trapping flow.
- Initiation of any one of the error trapping flows (ME, PE, or ICC).
- Use of the console LOAD switch.
- Use of the console RESET switch.

Interrupt Indicator (II)

The interrupt indicator and its associated console light are set ON by the I/O control when the selected peripheral terminates its operation and stores its S3 or S4 status character. This indicator is used in conjunction with the interrupt permit (IP) indicator to interrupt the normal program flow.

The interrupt indicator is set OFF when the interrupt trapping flow is entered, or when the console RESET switch is used.

Memory Error Indicator (ME)

The memory error indicator and its associated console light are set ON whenever a parity failure is detected in the data being read from memory. This flag is tested after every command execution; if it is ON, the processor normally enters the ME trapping flow. The only exception to this is when either the ME HALT switch or the EI indicator is ON; in either of these instances, the processor enters the halt state and all I/O terminates.

The memory error indicator is set OFF whenever the processor enters the ME trapping flow; however, if the processor enters the halt state, the memory error indicator can only be set OFF by the console RESET switch.

Program Error Indicator (PE)

The program error indicator and its associated console light are set ON: (1) whenever an attempt is made to access an illegal memory address, (2) whenever the repeat indicator (RI) is ON and the repeat counter is zero as the processor enters the repeat flow, or (3) whenever certain conditions are encountered in the MULTIPLY or DIVIDE commands (as stated below).

An illegal address is defined as follows:

- Any Mode 1 address that is not evenly divisible by four.
- Any Mode 1 or Mode 2 address equal to or greater than the physical memory size, except on systems with 64K memories (the 17th bit of the address is ignored on 64K memories, leaving a bit configuration that is less than 64K).
- Any Mode 2 address that addresses either one of the last two bytes of memory.

Improper programming with the hardware MULTIPLY and DIVIDE commands may also create a PE condition. For example, a PE condition is indicated for the multiply function if either the multiplier or the multiplicand is greater than eight bytes in length, or if any digit (other than the sign digit) of the A or B operand contains a value other than 0 through 9. A PE condition is indicated for the divide function if the effective A address is not evenly divisible by 4, if the divisor is equal to 0, if the divisor field is greater in length than the dividend field, if the divisor field is greater than eight bytes in length, if the quotient field is greater than eight bytes in length, or if the quotient will not fit into the assigned quotient field.

The PE flag is tested after every command execution; if it is ON, the processor normally enters the PE trapping flow. The only exception to this is when either the PE HALT switch or the EI indicator is ON. In either of these instances, the processor enters the halt state and all I/O terminates.

The program error indicator is normally set OFF whenever the processor enters the PE trapping flow; however, if the processor enters the halt state, the program error indicator can only be set OFF by the console RESET switch.

Illegal Command Code Indicator (ICC)

The illegal command code indicator is set ON whenever the processor encounters a command that is not included in the set of hardware commands available to the system. This situation occurs when a command code is invalid or when it is an interpretive.

An invalid command code is one that is neither contained in the hardware command code set nor recognized by the software as an interpretive. An interpretive command is a software command that usually represents a series of hardware commands to perform a specific function. For example, systems that are not equipped with the hardware MULTIPLY command use an interpretive command to perform the same function. When the processor encounters the interpretive, it sets the ICC indicator ON and traps out to a software routine that performs the multiplication by repeated use of the hardware ADD command.

The illegal command code indicator is set OFF by the command code trapping flow; it may also be set OFF by the console RESET switch.

Error Indicator (EI)

The error indicator and its associated console light are set ON by the ME or PE trapping flow to remember an error condition. If the error condition is repeated or a new error is encountered during the trapping flow, the error indicator causes the processor to enter the halt state.

The error indicator is set OFF either by the JUMP command (used at the start of the error recovery routine) or by the console RESET switch.

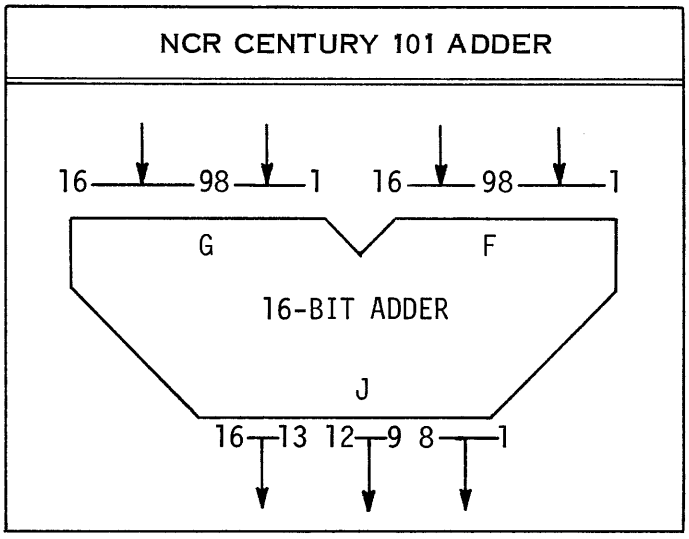
Read-Only-Memory Error Indicator (RE)

The read-only-memory error indicator and its associated console light are set ON whenever an error is detected in the ROM control logic. This flag is tested after every command execution; if it is ON, the processor enters either the ME trapping flow or a halt state, depending upon the state of the error indicator (EI). If EI is ON, the processor enters the halt state and all I/O terminates.

The RE indicator is normally set OFF as the processor enters the ME trapping flow; however, if the processor enters the halt state, the RE indicator can only be set OFF by the console RESET switch.

ALU ADDER

The arithmetic logic unit contains a 16-bit adder, which is capable of handling data as either 8- or 16-bit entries. Input to the adder is made at the G and F terminals; output is at the J terminal.



Data from memory is always input to the G terminal of the adder. This data may vary from 1 byte to 2 bytes in length, depending upon the function being performed. For example, during command setup, the command is input to the adder two bytes at a time, but during command execution, data is normally input one byte at a time.

The contents of the memory address register (L) may also be input to the G terminal of the adder. This data, which is always two bytes in length, is normally used to increment addresses in other hardware registers (CR, LA, LB, and LC).

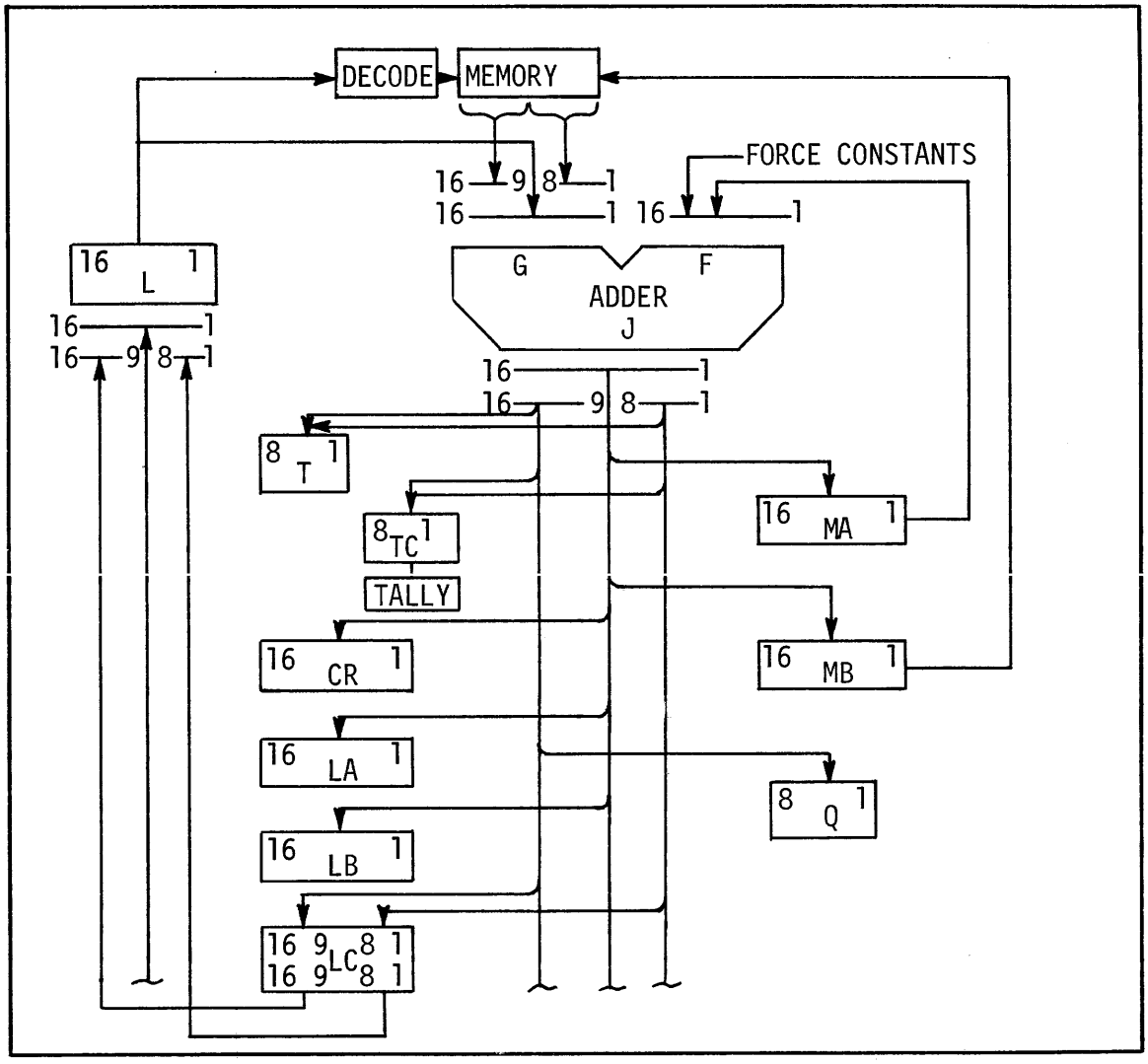
The F terminal of the adder is used to input data from the MA and MAP registers. This terminal is also used on certain commands to force constants (such as correction values for arithmetic functions and increment values for addresses) into the adder.

Output from the adder is available at the J terminal in 4-, 8-, or 16-bit increments. These outputs are placed into various hardware registers where they may be used to address memory or to control the operation (for example, 16-bit memory addresses are placed into the LA, LB, and CR registers where they can be incremented if necessary and used to address memory; 8- or 16-bit outputs are placed into the MB register where they are available for output to memory; the 8-bit command code is placed into the Q register where it is decoded and used to control the operation; 4-bit outputs are placed into the LC register and used to increment values in other registers, etc.).

Using the Adder for Command Setup

During the command setup phase, the adder transfers the command from memory to the appropriate hardware registers (two bytes at a time) and establishes the effective A and B addresses according to the addressing mode specified.

The following illustration indicates the hardware registers normally used during command setup and illustrates how the adder transfers information between them and memory.



All memory addressing is accomplished via the address register (L), which also circulates data through the G terminal of the adder for incrementing or decrementing purposes. For example, during command setup, the address of the command to be executed is moved from the sequence control register (CR) to the L register, where it is used to address memory; while memory is being accessed, the content of the L register is input to the G terminal of the adder, incremented by two, and loaded back into the CR register (the CR register now contains either the address of the next two bytes of the command, or the address of the next command in sequence).

As the command is input to the adder from memory, the command code is placed into the Q register where it can be checked for validity and decoded to determine the command length and its function.

The LA and LC registers are used during command setup to create an effective A address. As the command is read from memory, the address of the associated index register (if any) is stored in the LC register, and the partial A address (A2A1) is stored in the LA register. If an index register is specified, the content of LC is moved to the L register where it is used to address the associated index register in memory. While memory is being accessed, the content of the LA register is circulated through the G terminal of the adder (via the L register) to the MA register, where it is stored for later input to the F terminal of the adder. When the index register content is finally input to the G terminal of the adder, the content of MA is input to the F terminal; the two 16-bit values are added together, and the resulting effective A address is placed into the LA register.

The T value of the command, if any, is placed into the T register where it can be used during command execution. If a 4-byte command is being setup, the T value established during the previous 8-byte command remains unchanged.

The LB, LC, and MA registers are used to setup an effective B address for an 8-byte command in the same manner as an effective A address is created. If a 4-byte command is being setup, the address currently in the LB register (from the previous 8-byte command) is used as an implied B address for the 4-byte command (no further indexing is performed even though the previous 8-byte command may indicate indexing).

When incremental indexing (Mode 3) is specified, the MB register is used to write the incremented index register value back into memory. For example, after the effective A or B address is created, it is added to the content of the associated index register, and the result is placed into the MB register. This is accomplished by circulating the LA or LB register content through the adder and into the MA register, while the LC register is used to address the associated index register in memory. When the associated index register content is input to the G terminal of the adder, the MA register content is input to the F terminal, and the two 16-bit values are added together with the result being placed into the MB register. Because the LC register still contains the address of the associated index register, it is used to address memory again; the content of MB is then placed into the associated index register, thereby replacing the previous content with the incremented value.

When command setup is completed, the processor enters the execution phase to perform the function designated by the command code in the Q register. Adder functions (during the execution phase) are discussed as two separate topics, arithmetic adder functions and nonarithmetic adder functions. The first topic (arithmetic adder functions) explains the bit manipulation during the execution of the add, subtract, and compare commands; the second topic (nonarithmetic adder functions) explains the adder functions during execution of other commands such as move, pack, test, decode, etc.

Arithmetic Adder Functions

There are eight standard arithmetic commands for the NCR Century 101 Processor: ADD BINARY, ADD SIGNED, ADD UNSIGNED, SUBTRACT BINARY, SUBTRACT SIGNED, SUBTRACT UNSIGNED, COMPARE BINARY, and COMPARE SIGNED. The hardware MULTIPLY, DIVIDE, and LOGIC commands are optional and are explained later in this publication under the heading of Options.

All arithmetic commands binarily add the contents of one byte to the contents of another (subtract and compare commands perform their functions with complementary addition rather than actual subtraction). Because arithmetic operations are essentially add functions, only the four basic rules for binary addition need be considered:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0 With a carry

To perform an add function, the control logic inputs the A-field byte to the F terminal of the adder and the B-field byte to the G terminal. The two values are added together, with all carries being handled automatically at the end of the adder cycle.

00111101	Input to G Terminal (B Field)
00010110	Input to F Terminal (A Field)
00101011	Partial Sum (Without carries)
1111	Carries (Including carries caused by adding A to B)
01010011	Final Sum

The input data (binary or BCD), the data format (packed or unpacked), and the function to be performed (add, subtract, or compare) determine the method used to arrive at the result; therefore, each function is explained separately.

- Add Binary

  When executing an ADD BINARY command, the processor assumes all data to be binary and unsigned. Addition is performed one byte at a time, beginning with the rightmost byte in both fields (A and B). A carry from one byte is normally added to the next; however, a carry from the leftmost byte of the field is ignored. Consider the following example to add two 2-byte binary fields:

	2nd Byte	1st Byte	
	00111110	11101110	Input to G Terminal (B Field)
	11101000	10011001	Input to F Terminal (A Field)
	11010110	(1)01110111	Partial Sum
	1111 1	1111	Carries
Carry Ignored	(1)00100111	10000111	Final Sum

After command setup has been completed, the LA and LB registers contain the effective address of the leftmost byte of their respective fields, the Q register contains the command code, and the T register indicates the length of both fields. A tally (based on the T character) is created in the TC register to determine when the operation has been completed.

As the processor enters the execution phase, it computes the rightmost address of both fields by adding the contents of the T register to the contents of the LA and LB registers; the result of each addition is placed into the respective registers. (LA contains the rightmost address of the A field, and LB contains the rightmost address of the B field.)

The rightmost byte of the A field is accessed and placed into the MA register, where it is available for input to the F terminal of the adder. The content of LA is then decremented by one, pointing to the next byte of the A field. As the rightmost byte of the B field is accessed and input to the G terminal of the adder, the content of MA (A-field byte) is input to the F terminal. After addition, the output of the adder is placed into MB, where it is available for output to memory. A carry flag is set ON if a carry beyond bit 8 is detected.

Because LB has not been decremented yet, the content of MB (final sum) is placed into the rightmost byte of the B field. LB is then decremented by one to point to the next byte of the B field. The tally count (TC) is also decremented by one.

When TC has been decremented to zero, the command terminates and the bit 8 carry (if any) is ignored (the overflow flag is not set ON); LA and LB have both been decremented to their original values. If TC is not equal to zero, processing continues with the next two bytes (A and B); a carry beyond bit 8 of the previous two bytes is added to the partial sum of the next two bytes.

- Subtract Binary

When executing a SUBTRACT BINARY command, the processor assumes all data to be unsigned binary data. The subtract function is performed one byte at a time, beginning with the rightmost byte of both fields (A and B). A carry from one byte normally affects the next; however, a carry from the leftmost byte of the field is ignored.

Binary subtraction is performed as complementary addition; that is, the A-field content is complemented (all 1 bits are set to zero, and all zero bits are set to 1) and added to the B-field content. An initial carry is forced into the first bit position of the rightmost byte to arrive at a true two's complement. Consider the following example to subtract two 2-byte binary fields. Assume that the B field contains 01100011 11001101 (25,549 decimal), and the A field contains 00011000 01010001 (6,225 decimal).

```
                2nd Byte          1st Byte
                01100011          11001101   Input to G (B Field)
                11100111          10101110   Input to F (1's Complement of A Field)
                10000100        (1)01100011  Partial Sum
                11  1111                1111 Carries
                                           1 Initial Carry For 2's Complement
            ___________          ________
Carry ←(1)01001011                01111100   Final Sum
Ignored
```

After command setup has been completed, the LA and LB registers contain the effective address of the leftmost byte of their respective fields, the Q register contains the command code, and the T register indicates the length of both fields. A tally (based on the T character) is created in the TC register to determine when the operation has been completed.

As the processor enters the execution phase, it computes the rightmost address of both fields by adding the contents of the T register to the contents of the LA and LB registers; the result of each addition is placed into the respective registers. (LA contains the rightmost address of the A field, and LB contains the rightmost address of the B field.)

The rightmost byte of the A field is input to the adder, complemented, and placed into the MAP register, where it is available for input to the F terminal of the adder. The content of LA is then decremented by one, pointing to the next byte of the A field. As the rightmost byte of the B field is accessed and input to the G terminal of the adder, the content of MAP (1's complement of the A-field byte) is input to the F terminal. The two inputs are added, an initial carry is forced (for the 2's complement), and the final sum is placed into the MB register, where it is available for output to memory. A carry flag is set if a carry beyond bit 8 is detected.

Because LB has not been decremented yet, the content of MB (final sum) is placed into the rightmost byte of the B field. LB is then decremented by one to point to the next byte of the B field. The tally count (TC) is also decremented by one.

When TC has been decremented to zero, the command terminates and the bit 8 carry (if any) is ignored (the overflow flag is not set ON); LA and LB have both been decremented to their original values. If TC is not equal to zero, processing continues with the next two bytes (A and B); a carry beyond bit 8 of the previous two bytes is added to the partial sum of the next two bytes.

- Compare Binary

The COMPARE BINARY command functions exactly the same as the SUBTRACT BINARY command, except that the final sum is not output to memory and the appropriate comparison flag (less, equal, or greater) is set ON.

During the last adder cycle (cycle in which TC is decremented to zero), the processor sets all comparison flags OFF, then checks the output of the adder and the bit 8 carry flag to determine which comparison flag to set ON. If the adder output equals zero (indicating that A is equal to B in every

subtraction cycle), the equal flag (E) is set ON. If the adder output is not equal to zero, the processor checks the bit 8 carry flag to determine whether A is less than or greater than B. The bit 8 carry flag being ON indicates that A is less than B, and the less flag (L) is set ON; the bit 8 carry flag being OFF indicates that A is greater than B, and the greater flag (G) is set ON.

- Adding Signed Packed Decimal Data

When executing the ADD SIGNED command, the processor assumes all data to be signed, packed, decimal information; that is, each 8-bit byte (except the rightmost byte of the field) is expected to contain two BCD digits with binary values ranging from 0 through 9. The rightmost byte of both fields (A and B) is expected to contain one 4-bit BCD character in bits 8-5 and an arithmetic sign in bits 4-1 (a negative field is indicated by the bit configuration 1101, a positive field by any other configuration). Addition is performed from right-to-left one byte (8-bits) at a time, beginning with the rightmost byte of both fields. The sign bits are not added, but are used to determine the method to be used in arriving at the result.

- Adding Fields With Like Signs

If the signs of both fields are equal, the processor adds the contents of the A field to the contents of the B field and places the result into the B field without changing the sign. When performing BCD addition on fields with like signs, the processor automatically adds an excess 6 (0110) to each A-field BCD character, then adds the adjusted A-field byte to the corresponding B-field byte. This is an attempt to prevent generating illegal BCD codes (bit configurations greater than nine).

The following example illustrates the addition process for adding one A-field byte (two BCD characters) to one B-field byte. The example on the left shows two illegal BCD codes as a result of normal BCD addition; the example on the right shows how the excess 6 is used to prevent illegal BCD codes.

BCD ADDITION WITHOUT EXCESS 6	BCD ADDITION WITH EXCESS 6
0011 0011 A-Field Byte (BCD 3,3) 1000 1000 B-Field Byte (BCD 8,8) 1011 1011 Illegal BCD Codes	0011 0011 A-Field Byte (BCD 3,3) 0110 0110 Excess 6 0101 0101 11 11 Carries 1001 1001 Adjusted A-Field Byte 1000 1000 B-Field Byte (1)0001 (1)0001 11 Carries 0010 0001 Final Sum A carry beyond the leftmost bit of the sum is added to the next byte; a carry beyond the leftmost byte of a field is used to set the overflow flag ON.

In some cases, the excess six adjustment is not needed and actually causes an illegal bit configuration or an incorrect result. To prevent either case from being output to memory, the processor checks the output of the adder for a carry from one 4-bit BCD sum to the next; if there is no carry for a particular 4-bit sum, the logic circuitry adds a binary ten correction (1010) to that sum, effectively subtracting the excess 6.

The following example shows the binary ten correction of an incorrect result. Only the rightmost four bits of the example needs to be corrected; the leftmost four bits are accepted as a legal BCD value because they generated a carry to the next byte or to the overflow flag.

BCD ADDITION WITH EXCESS 6 AND BINARY 10 CORRECTION

	0001	0001	A-Field Byte (BCD 1,1)
	0110	0110	Excess 6
	0111	0111	Adjusted A-Field Byte
	1001	0001	B-Field Byte (BCD 9,1)
	1110	0110	
	111	111	Carries
To Next Byte (1) or Overflow Flag	0000	1000	Incorrect Result
		1010	Ten Correction
	0000(*)	0010	Final Sum

* Carries between BCD characters or between bytes are ignored during the ten correction phase.

- Adding Fields With Unlike Signs

If the signs of the A and B fields are unequal, the processor complements each A-field byte before adding it to the corresponding B-field byte. Complementary addition effectively subtracts the contents of one byte from another. Because illegal BCD codes may be generated during the addition process, the processor checks for carries from one BCD sum to another and from one byte to another; the logic circuitry automatically performs a binary ten correction on any 4-bit sum that does not generate a carry to the next 4-bit sum or to the next byte.

A carry beyond the leftmost byte of the result indicates that the absolute A value is less than or equal to the absolute B value, and the final sum is a true value; no carry beyond the leftmost byte indicates that the absolute A value is greater than the absolute B value, and the intermediate sum is actually the tens complement of the true sum (this sum is called an intermediate sum because the processor recomplements this value before storing it in memory).

The following example indicates the procedure for adding two fields with unlike signs. The left-hand example shows a 2-byte A field with a positive sign being added to a 2-byte B field with a negative sign (only three BCD characters are shown for each field because the sign bits are not added). Because the absolute value of the A field is smaller than

the absolute value of the B field, a carry is generated beyond the leftmost byte of the result, indicating a true sum. The right-hand example shows a 2-byte A field with a negative sign being added to a 2-byte B field with a positive sign (again, only three BCD characters are shown because the sign bits are not added). Because the absolute value of the A field is greater than the absolute value of the B field, no carry is generated beyond the leftmost byte of the result, indicating an intermediate sum rather than a final sum.

BCD ADDITION WITH UNLIKE SIGNS A ≤ B	BCD ADDITION WITH UNLIKE SIGNS A > B
A Field = 154+ (0001 0101 0100 +) B Field = 456- (0100 0101 0110 -)	A Field = 456- (0100 0101 0110 -) B Field = 154+ (0001 0101 0100 +)
1110 1010 1011 A Field (1's Comp.) 0100 0101 0110 B Field **1010** 1111 1101 1 1 1111 111 Carries 1 Initial Carry (for 2's complement) **(1)0011(1)0000(1)0010 Final Sum***	1011 1010 1001 A Field (1's Comp.) 0001 0101 0100 B Field 1010 1111 1101 11 1 Carries 1 Initial Carry (for 2's complement) (0)1100(0)1111(0)1110 Invalid BCD Code** 1010 1010 1010 Ten Correction 0110 0101 0100 11 11 Carries 0110 1001 1000 Intermediate Sum
* Because each 4-bit sum generates a carry to the next 4-bit sum, no ten correction is necessary. A carry beyond the leftmost bit of the field indicates that A ≤ B; the final sum is the true sum.	** Because no carry is detected between any 4-bit sum, a ten correction is required for each one. No carry beyond the left most bit of the field indicates that A > B; the intermediate sum is the 10's complement of the true sum.

If the sum of the addition (after any ten correction) is the true sum, the processor terminates the command, leaving the sign of the B field unchanged. If the sum of the addition indicates a tens complement of the true value, the processor recomplements the entire B field, which currently contains the intermediate sum. The recomplemented sum is then placed back into memory, and the sign of the B field is changed. The following example indicates the process used to recomplement the intermediate sum.

```
(0110   1001   1000  Intermediate Sum)
 1001   0110   0111  1's Complement
                  1  Initial Carry (for 2's Complement)
               111   Carries
 1001   0110   1000  Incorrect Result
 1010   1010   1010  Ten Correction
 0011   1100   0010
     (1)11           Carries (Carries between bytes are
                     ignored during ten correction.)
 0011   0000   0010  True Sum
```

After command setup has been completed, the LA and LB registers contain the effective address of the leftmost byte of their respective fields, the Q register contains the command code, and the T register indicates the length of both fields. A tally (based on the T character) is created in the TC register to determine when to terminate the command.

The first step of the execution phase adds the content of the T register to both the LA and LB registers, effectively computing the rightmost addresses of the A and B fields. The next step tests the signs of the two fields; a control flag is set ON, if the signs are unequal (opposite).

If the control flag is ON (indicating opposite signs), the first A-field byte is complemented (all 1 bits are set to 0 and all 0 bits are set to 1) as it is read from memory and placed into the MAP register. If the control flag is OFF (indicating like signs), a binary 6 (0110) is added to the A-field byte and the adjusted result is placed into the MA register. The content of the LA register is decremented by one, pointing to the next A-field byte. The first B-field byte is then accessed and added to either the complemented A-field byte (MAP) or to the adjusted A-field byte (MA); the result is placed into the MB register.

Before the MB register is output to memory, the processor tests the b8 and b4 carry flags to determine whether any binary ten correction is needed. If no carry is indicated by either of these flags, the logic circuitry performs the binary ten correction (where necessary) and places the result back into the MB register; the content of MB is then output to memory at the address indicated by the LB register. After the content of MB has been placed into memory, LB and the tally count are decremented by one. This process is repeated until the tally count has been decremented to zero.

When the tally count equals zero, the processor checks the control flag to determine whether the signs are equal or unequal. If the signs are equal (control flag OFF), the processor terminates the command. If the signs are unequal (control flag ON), the processor checks the b8 carry flag to determine whether the new B-field content is a true sum or an intermediate sum. If the bit 8 carry flag is ON (indicating a true sum), the processor terminates the command; otherwise, the processor computes the rightmost address of the B field, then complements that field and stores it back into memory with the appropriate sign. If the sum is a negative value, the sign is set to 1101; otherwise, it is set to 1011.

- Subtracting Signed, Packed, Decimal Data

The processor handles signed, packed, decimal information with the SUBTRACT SIGNED command in the same manner in which it handles signed, packed, decimal data with the ADD SIGNED command; the only exceptions being as follows:

- When subtracting fields with unlike signs, the processor performs the same functions as in adding fields with like signs. For example, an excess 6 is added to each A-field byte as it is read from memory, then the adjusted A-field byte is added to the corresponding B-field byte. Binary ten corrections are made where necessary.

- When subtracting fields with like signs and the absolute value of the A field is greater than the absolute value of the B field, the processor performs the same functions as in adding fields with unlike signs when the absolute value of the A field is less than or equal to the absolute value of the B field. For example, the A field is complemented and added to the B field; binary ten corrections are made where necessary. The result of the addition (after ten correction) is the true sum.

- When subtracting fields with like signs and the absolute value of the A field is less than or equal to the absolute value of the B field, the processor performs the same functions as in adding fields with unlike signs when the absolute value of the A field is greater than the absolute value of the B field; binary ten corrections are made where necessary. The intermediate sum (tens complement of the true sum) is recomplemented to arrive at the true sum, and the sign of the B field is changed to match that of the A field.

- Comparing Signed, Packed, Decimal Data

The COMPARE SIGNED command functions the same as the COMPARE BINARY command, except that the COMPARE SIGNED command also tests the signs of both fields. The processor compares each byte of the A field to a corresponding byte of the B field, beginning with the rightmost byte of both fields. The sign of the A field is compared to the sign of the B field and, if the signs are equal, a control flag is set ON. Comparison is then made on the data one byte at a time and a tally count, which is initially set according to the length of the A and B fields, is decremented by one after each comparison. When the tally count is equal to zero, the processor tests the control flag and the output of the adder.

If the control flag is ON (indicating that the signs are equal), the processor checks both the output of the adder and the bit 8 carry flag to determine the result of the operation:

- An adder output other than zero and the bit 8 carry flag ON indicates that the absolute value of the A field is less than the absolute value of the B field; therefore, the less flag (L) is set ON.

- An adder output equal to zero and the bit 8 carry flag ON indicates that the absolute value of the A field is equal to the absolute value of the B field; therefore, the equal flag (E) is set ON.

- An adder output other than zero and the bit 8 carry flag OFF indicates that the absolute value of the A field is greater than the absolute value of the B field; therefore, the greater flag (G) is set ON.

- If the control flag is OFF (indicating that the signs are not equal) and the B field has a negative sign, the processor sets either the greater (G) or less (L) flag ON, providing that both fields are not equal to zero (if both fields are equal to zero, the processor sets the equal flag ON even though the signs are not equal).

Comparison of data characters is accomplished by complementary addition; that is, the processor complements each byte of the A field before adding it to the corresponding byte of the B field. The output of the adder is tested on every cycle and the bit 8 carry flag is tested when the last byte of the A field is being compared to the last byte of the B field. No corrections are made and no output is made to memory.

After command setup has been completed, the LA and LB registers contain the effective address of the leftmost byte of their respective fields, the Q register contains the command code, and the T register indicates the length of both fields. A tally (based on the T character) is created in the TC register to determine when to terminate the command.

The first step of the execution phase adds the content of the T register to both the LA and LB registers, effectively computing the rightmost addresses of the A and B fields. The next step tests the signs of the two fields; a control flag is set ON if the signs are equal.

The first A-field byte is complemented (all 1 bits are set to 0 and all 0 bits are set to 1) as it is read from memory and placed into the MAP register. The first B-field byte is then accessed and added to the complemented A-field byte; the result is placed into the MB register, but will not be output to memory. The LA and LB registers are decremented by one, thereby pointing to the next byte of their respective fields. The tally count is also decremented by one and tested to see if it equals zero; if it is not equal to zero, the processor accesses the next A-field byte and repeats the above process.

When the tally count equals zero, the processor sets the comparison flags according to the criteria explained earlier. The command terminates with the LA and LB registers containing their original values. Because the content of the MB register is not output to memory during the execution phase, the A and B fields each contain their original values.

- Adding Unsigned, Unpacked Decimal Data

When executing the ADD UNSIGNED command, the processor assumes that the rightmost four bits of each byte contain a legal BCD code (4-bit binary configuration equal to or less than nine); the leftmost four bits, which usually contain the ASCII zone bits, are ignored and, therefore, may contain any bit configuration. Because the processor ignores the leftmost four bits of each byte during the addition process, it places the ASCII bit configuration 0011 into the leftmost four bits of the result before placing the result into memory.

Addition is performed one byte at a time, beginning with the rightmost byte of both fields (A and B). A carry beyond bit 4 of one byte is added to the byte on its left, except in the leftmost byte of the result; a carry beyond bit 4 of that byte sets the overflow flag ON, indicating that the result is larger in size than the B field.

Addition is accomplished in the same manner as with the ADD SIGNED command using like signs; the only exception is that each 8-bit byte of unsigned, unpacked, decimal data contains only one 4-bit BCD data character instead of two (the sign characters are ignored during the add process, and the processor inserts the ASCII 0011 bit configuration into the leftmost four bits of the result before storing it in memory). The processor begins the add function by adding an excess 6 to each A-field byte, then adding the adjusted A-field byte to a corresponding B-field byte. If the result of this addition does not generate a carry beyond b4, a binary ten correction is made. The following example illustrates both cases (one without the binary ten correction, the other with the binary ten correction).

DECIMAL ADDITION WITHOUT BINARY TEN CORRECTION	DECIMAL ADDITION WITH BINARY TEN CORRECTION
A-Field Byte 0011 0001 (ASCII 1) B-Field Byte 0011 1001 (ASCII 9)	A-Field Byte 0011 0001 (ASCII 1) B-Field Byte 0011 0001 (ASCII 1)
0001 A-Field Byte (ASCII 1) 0110 Excess 6 0111 Adjusted A-Field Byte 1001 B-Field Byte (ASCII 9) 1110 111 Carries 0011(1)0000 Final Sum	0001 A-Field Byte (ASCII 1) 0110 Excess 6 0111 Adjusted A-Field Byte 0001 B-Field Byte (ASCII 1) 0110 111 Carries (0)1000 1010 Ten Correction 0011 0010 Final Sum
The carry beyond bit 4 is either added to the next byte or used to set the overflow flag ON; no binary ten correction is required.	The absence of a carry beyond bit 4 causes a binary ten correction, which effectively subtracts the excess 6.

After command setup has been completed, the LA and LB registers contain the effective address of the leftmost byte of their respective fields, the Q register contains the command code, and the T register indicates the length of both fields. A tally (based on the T character) is created in the TC register to determine when to terminate the command.

The first step of the execution phase adds the content of the T register to both the LA and the LB registers, effectively computing the rightmost addresses of the A and B fields. The processor then accesses the rightmost byte of the A field, adds a binary 6 (0110) to its content, and places the result into the MA register, where it is available for input to the F terminal of the adder. The LA register is decremented by one and points to the next byte of the A field. As the processor accesses the rightmost byte

of the B field, the control logic inputs the content of the MA register (adjusted A-field byte) into the F terminal of the adder and the B-field byte into the G terminal; the two values are added, and the result is placed into the MB register.

Before placing the content of the MB register into memory, the processor tests the b4 carry flag to determine whether a binary ten correction is needed; if no carry is indicated, the logic circuitry performs the binary ten correction and places the result back into the MB register. The processor then inserts the ASCII zone bits (0011) into the leftmost four bits of the byte in MB and outputs the 8-bit byte to memory at the location addressed by the LB register. The content of LB and the tally count are decremented by one. If the tally count has been decremented to zero, the processor terminates the command; otherwise, the above process is repeated until the tally count is equal to zero.

- Subtracting Unsigned, Unpacked, Decimal Data

When executing the SUBTRACT UNSIGNED command, the processor assumes that the rightmost four bits of each byte contains a legal BCD code (4-bit binary configuration equal to or less than nine); the leftmost four bits, which usually contain the ASCII zone bits, are ignored and, therefore, may contain any bit configuration. Since the processor ignores the leftmost four bits of each byte during the operation, it places the ASCII bit configuration 0011 into the leftmost four bits of the result before placing the result into memory.

Subtraction is accomplished by complementary addition; that is the content of each A-field byte is complemented (all 1 bits are set to 0 and all 0 bits are set to 1) before it is added to the corresponding B-field byte. Complementary addition is performed one byte at a time, beginning with the rightmost byte of both fields (A and B). A carry beyond bit 4 of one byte is added to the byte on its left, except in the leftmost byte of the result; a carry beyond bit 4 of that byte is dropped. If the result of the addition does not generate a carry beyond bit 4, a binary ten correction is made. The following example illustrates both cases (one without the binary ten correction, the other with the binary ten correction).

COMPLEMENTARY ADDITION WITHOUT BINARY TEN CORRECTION			COMPLEMENTARY ADDITION WITH BINARY TEN CORRECTION		
A-Field Byte [0011] 0001 (ASCII 1) B-Field Byte [0011] 0100 (ASCII 4)			A-Field Byte [0011] 0100 (ASCII 4) B-Field Byte [0011] 0001 (ASCII 1)		
	1110	1's Complement A-Field Byte		1011	1's Complement A-Field Byte
	0100	B-Field Byte		0001	B-Field Byte
	1010			1010	
	1	Carries		11	Carries
	1	Initial Carry (2's Comp.)		1	Initial Carry (2's Comp.)
	(1)0011			(0)1101	
				1010	Binary Ten Correction
0011	0011	Final Sum (ASCII 3)	0011	0111	Final Sum (ASCII 7)

If the absolute value of the A field is greater than the absolute value of the B field, the final sum (after the binary ten correction) is actually the tens complement of the true sum; it is this complement that will be stored in memory.

After command setup has been completed, the LA and LB registers contain the effective address of the leftmost byte of their respective fields, the Q register contains the command code, and the T register indicates the length of both fields. A tally (based on the T character) is created in the TC register to determine when to terminate the command.

The first step of the execution phase adds the content of the T register to both the LA and LB registers, effectively computing the rightmost addresses of the A and B fields. The processor then accesses the rightmost byte of the A field, complements the rightmost four bits (ignoring the leftmost four bits), and places the result into the MAP register, where it is available for input to the F terminal of the adder; LA is decremented by one and points to the next byte of the A field. As the processor accesses the rightmost byte of the B field, the control logic inputs the content of the MAP register (complemented A-field byte) into the F terminal of the adder, and the B-field byte is input to the G terminal, thereby adding the complemented A-field byte to the B-field byte. The result is placed into the MB register.

Before placing the MB register content into memory, the processor tests the b4 carry flag to determine whether a binary ten correction is needed; if no b4 carry is detected, the logic circuitry performs the binary ten correction. The processor places the ASCII zone bits (0011) into the leftmost four bits of the byte in MB, then outputs the 8-bit byte to memory at the location addressed by the LB register. After placing the result into memory, the processor decrements the LB register and tally count by one (the LB register points to the next byte of the B field). If the tally count has not been decremented to zero, the processor repeats the above process; however, when the tally count equals zero, the processor terminates the command. Because LA and LB are decremented during each step of the execution phase, they will contain their original values when the command terminates.

Nonarithmetic Adder Functions

The adder performs many functions other than those normally associated with the arithmetic commands; for example, the adder is used to perform bit or character comparisons, to move, pack or unpack data, to jump or branch to other commands, etc. The following discussion explains the adder functions for various nonarithmetic commands; each explanation assumes that the command setup phase has been completed.

- Using the Adder to Move Data

  A field of data (up to 256 bytes) can be moved one byte at a time from the A field to the B field, or from the B field to the A field. If the B field is being moved, the processor always begins with the rightmost byte of each field; if the A field is being moved, the processor may begin with either the leftmost byte or the rightmost byte.

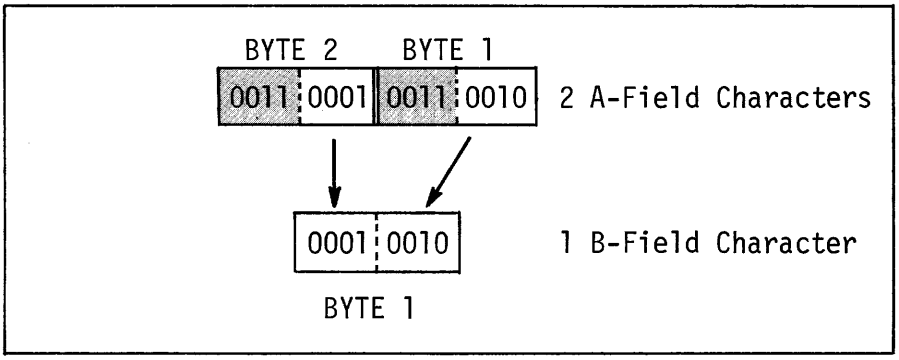
After command setup, the T register contains the length of both fields, the LA and LB registers contain the leftmost address of their respective fields, and the Q register contains the command code. The processor examines the command code to determine the direction of the move (left-to-right or right-to-left) and the operand to be moved; it also sets up a tally according to the value in the T register.

If the move is to occur from left-to-right, the adder simply addresses each field (according to the effective addresses in LA and LB) and transfers the data from the A field to the B field, via the MB register. As each byte is moved, the adder decrements the tally and increments the LA and LB registers; when the tally has been decremented to zero, the command terminates with the LA and LB registers each pointing to the next available byte of memory.

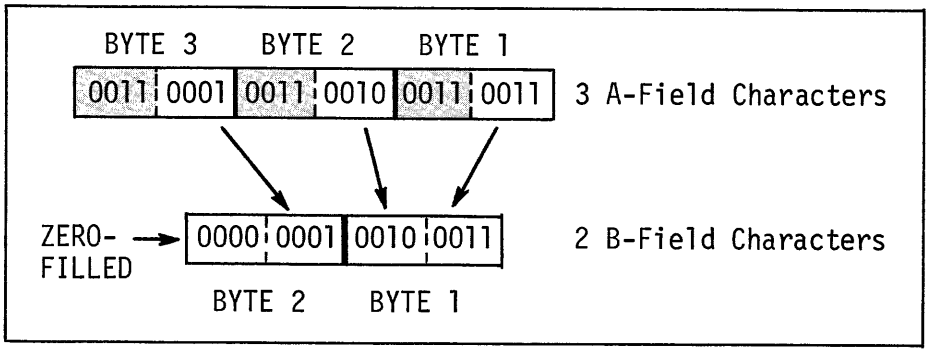
If the move is to occur from right-to-left, the adder computes the rightmost address of both fields by adding the value in the T register to the values in the LA and LB registers; the result of each addition is placed back into its respective register. The transfer of data (from A-to-B or from B-to-A) is accomplished one byte at a time, via the MB register. As each byte is moved, the adder decrements both the tally and the contents of the LA and LB registers; when the tally has been decremented to zero, the command terminates with the LA and LB registers each containing their original values.

- Using the Adder to Pack Data

A field of NCR Century (ASCII) characters can be compressed to half or approximately half of its original size by dropping the zone bits of each byte and placing the remaining four bits into another field as BCD characters (two 4-bit data characters per byte). Packing is performed sequentially from left-to-right, beginning with the leftmost A- and B-memory locations; each A-field character is stripped of its zone bits and placed into the appropriate position (b8-b5 or b4-b1) of a corresponding B-field character.



If the A field contains an uneven number of bytes, the leftmost four bits of the B field are zero-filled.



After command setup, the T register contains the length of the A field (0-255, with 0 equal to 256), the LA and LB registers contain the leftmost address of their respective fields, and the Q register contains the command code. During command execution, the processor sets up a tally according to the T value and, if the T value is an uneven number, sets a control flag ON.

Each 8-bit B-field character is constructed (four bits at a time) in the MB register, then transferred to memory according to the address in the LB register; each time the MB register is output to memory, the LB register is incremented by one, thereby pointing to the next available B-field position.

The first byte of the A field, which is addressed by the contents of LA, is input to the G terminal of the adder; the four zone bits (b8-b5) are dropped, and the four digit bits (b4-b1) are placed into the MB register either at positions b8-b5 or at positions b4-b1 (if the control flag is ON, the adder places zeros into b8-b5 and the first A-field character into b4-b1). As each A-field character is accessed, the LA register is incremented by one, and the tally counter is decremented by one; the command terminates when the tally is decremented to zero. When the command execution is completed, LA and LB each point to the next available byte of memory.

- Using the Adder to Unpack Data

Packed data (consisting of two 4-bit digits per byte) can be unpacked and restored to its original ASCII configuration by assigning the appropriate zone bits to each digit. The hardware UNPACK command restores the following characters to their original ASCII values; other characters may be unpacked with this command, but the proper zone bits will not be assigned.

0	1	2	3	4	5	6	7	8	9	*	+	,	-	.	/

- Using the Adder to Decode Data

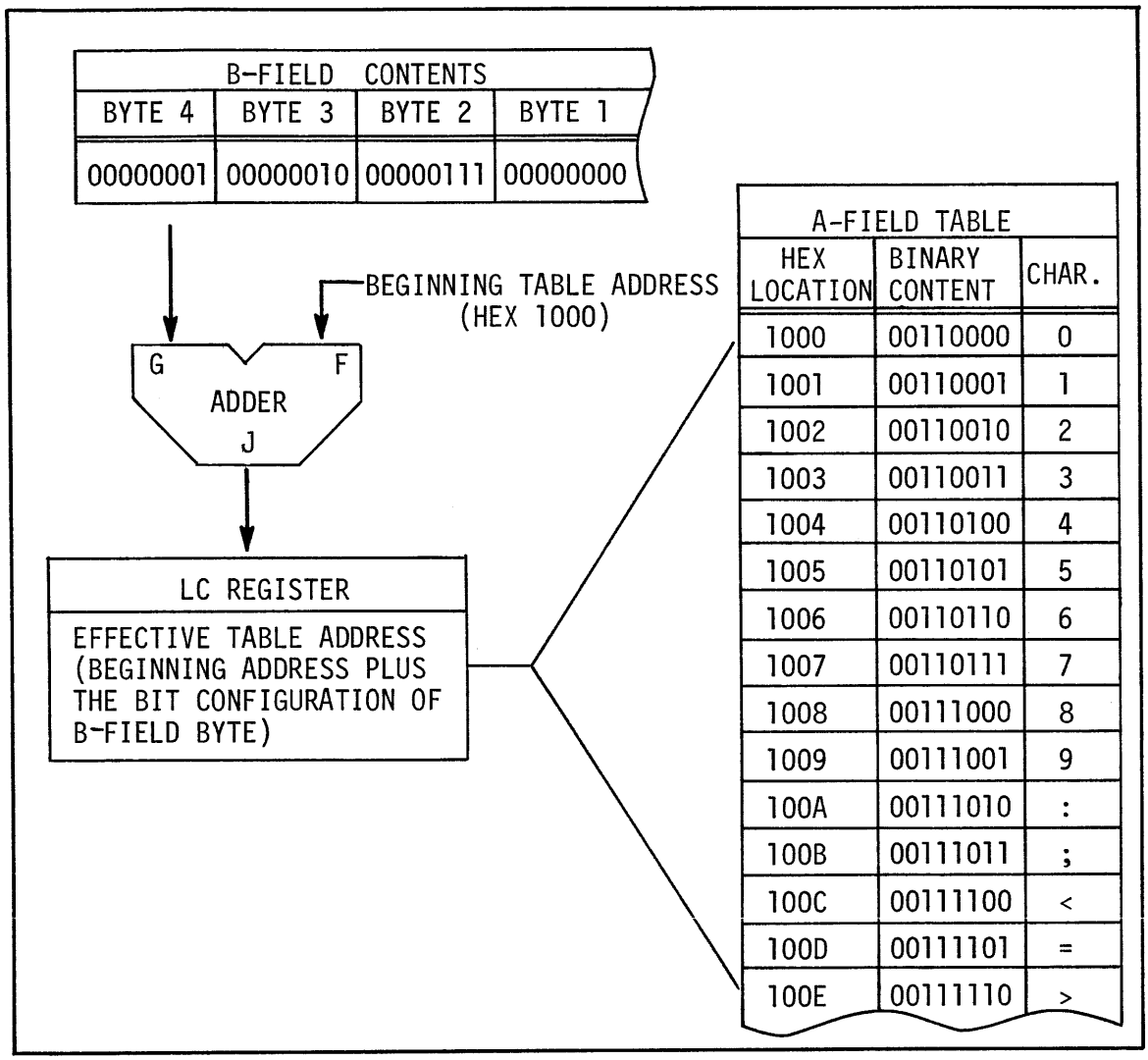
Two hardware commands (DECODE ALL and DECODE TO DELIMITER) enable the processor to convert data in memory from one code to another without using lengthy software routines. For example, either of the decode commands may be used to convert punched card codes, which have been read into the input buffer in memory, directly to NCR Century ASCII codes for storage and internal use, or they may be used to convert NCR Century ASCII codes to punched card codes before placing the data into the output buffer. Decoding is accomplished by a table lookup procedure. The A field contains a prearranged table, consisting of an appropriate replacement character for each character of the code set to be decoded. The B field contains the data to be decoded. The B field is read from left-to-right, one byte at a time.

As each byte of the B field is read, its content is used to create an effective address in the LC register, which points to the desired replacement character in the A-field table. The replacement character is then accessed and placed into the B-field byte that selected it. This procedure is repeated for every byte in the B field, unless the DECODE TO DELIMITER command is used and a delimiter character (any byte with bit 8 ON) is encountered in the table; the delimiter character may terminate the operation before the entire B field is decoded.

If the decode command terminates on a delimiter character, that character is not transferred to the B-field byte and the hardware equal flag (E) is set ON. If the decode command does not encounter a delimiter character before the entire B field is decoded, the greater flag (G) is set ON. These flags may be used by the hardware BRANCH commands to transfer control to different routines, depending upon their status.

When command setup has been completed, the T register contains the length of the B field (0-255, with 0 equal to 256), the LA and LB registers contain the leftmost address of their respective fields, and the Q register contains the command code. The processor sets up a tally count equal to the T value; this tally is used to determine when the entire B field has been decoded.

At the start of command execution, the content of LA (beginning address of the table) is placed into the MA register, and the first byte of the B field is accessed according to the address in LB. As the B-field byte is input to the G terminal of the adder, the content of MA is input to the F terminal; the two values are added together, creating an effective address which is then placed into the LC register. This address is used to access the desired replacement character in the decode table. The following illustration shows how each B-field byte is used to address an item in the A-field table; once the item has been located, its content replaces the B-field byte that selected it.

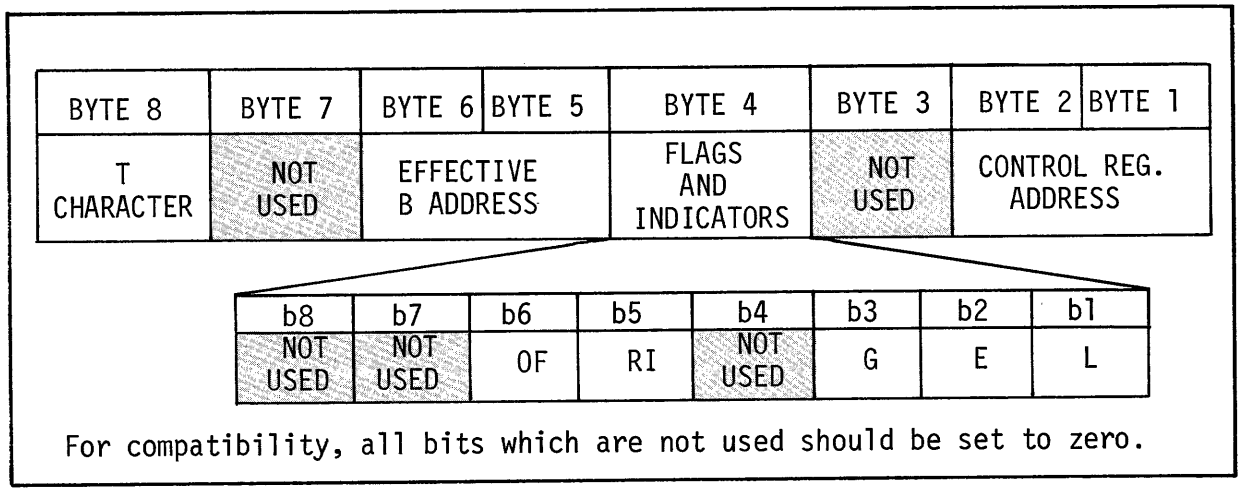


The replacement character is read from the table and placed into the MB register where bit 8 is checked. If bit 8 is ON and the DECODE TO DELIMITER command is being used, the content of MB (delimiter character) is not placed into the B field. The command terminates after decrementing the tally by one and setting the equal flag (E) ON (E is set ON even if this is the last byte to be accessed for the B field); the LB register is not incremented and, therefore, points to the address just beyond the last decoded B-field character. If bit 8 is OFF, the content of MB is transferred to the memory location addressed by LB, the tally is decremented by one, LB is incremented by one, and decoding continues with the next B-field byte; when the tally is decremented to zero, the processor sets the greater flag (G) ON and terminates the command.

Upon termination of the command, LA still points to the beginning address of the table; LB points one byte beyond the last decoded B-field byte (this may be the byte that selected the delimiter character or the next byte of memory beyond the B field, depending upon the method of termination). The last logic flow of the command places the content of LB into the Next-Address Index Register (IR9) at memory location 0026 (hex), which may be used by a subsequent software routine to determine the address of the last successfully decoded B-field character.

- Using the Adder to Establish a New Processor Status

The RESTORE command enables the processor to establish a new T value, a new effective B address, a new sequence control address, and new settings for the OF, RI, G, E, and L flags and indicators. This is accomplished by accessing an 8-byte field previously placed into memory by the user's program. This field must contain the following types of information and be in the format indicated; in addition, the beginning byte (byte 8) must be located at a 0 modulo 4 address (address evenly divisible by 4).



The RESTORE command is normally a 4-byte command, consisting of a command code and a 3-byte A operand. The effective A address created from the A operand points to the leftmost byte of the pre-established 8-byte field. During command execution, the 8-byte field is accessed two bytes at a time, and the contents of each pair of bytes are placed into the appropriate hardware registers or used to set the appropriate hardware flags and indicators. If no errors are detected during execution of this command, control is given to the command addressed by the control register (this address must also be 0 modulo 4). The RESTORE command is not repeated even if the repeat indicator was ON before command setup.

After command setup, the Q register contains the command code, and the LA register contains the effective address of the 8-byte field. During the execution phase, the first two bytes of the 8-byte field are accessed and input to the G terminal of the adder; the T value is placed into the T and TC registers, and the second byte, which is usually zero-filled, is ignored. The content of LA is incremented by two, thereby enabling the processor to access the next two bytes (bytes 6 and 5). The 16-bit address contained within these two bytes is placed into the LB register, where it can be used as an implied B address by a subsequent 4-byte command if desired. Again, the content of LA is incremented by two, pointing to the next two bytes (bytes 4 and 3). The content of byte 4 is used to set the related hardware flag or indicator ON or OFF, depending upon the related bit values; 0 sets the related flag or indicator OFF, 1 sets it ON. Byte 3, which is usually zero-filled, is ignored. LA is incremented by two, pointing to the last two bytes of the field (bytes 2 and 1). The 16-bit address contained in these two bytes is placed into the sequence control register (CR) where it is checked for validity (0 modulo 4). If the content of CR is valid, it is used to address the next command to be executed; if it is invalid, a PE condition results.

- Using the Adder to Setup a Repeat or Skip Function

The REPEAT command enables the programmer to either perform the command immediately following it a predetermined number of times, or to skip that command and proceed with the next command in sequence. A binary value, which is referenced by the A operand of the REPEAT command, determines the number of times that the following command is to be executed; if this binary value is equal to zero, the following command is not executed at all, instead control is given to the command that follows it in sequence.

If the binary value is greater than zero, that value is transferred to the repeat counter at memory location 0020 (hex), and the repeat indicator (RI) is set ON; the processor then terminates the REPEAT command and continues with the command to be repeated. Since the repeat indicator is ON, the processor enters the repeat trapping flow immediately after the execution phase of the command being repeated. The repeat flow tests the repeat counter for zero (it may have been set to zero by the command being repeated); if it is equal to zero, the processor immediately indicates a program error (PE). If the repeat counter is not equal to zero on this first test, it is decremented by one and tested again for zero; if it is not equal to zero, the processor decrements the sequence control register (CR) by the length of the command being repeated and returns control to that command. This process continues until the repeat counter is decremented to zero; at that time, the repeat indicator is set OFF and control is given to the next command in sequence.

An explanation of the repeat flow is given in this publication under "Trapping Flows." The following explanation considers only the setup of the repeat counter, the control of the repeat indicator, and the method used to skip the following command when the repeat counter is equal to zero.

When the REPEAT command setup phase is completed, the LA register contains the effective A address of a 1-byte binary repeat value, which has been preestablished in memory by the user's program. During the execution phase, the 1-byte field is accessed and its content is input to the G terminal of the adder. The output of the adder is then placed into the MB register where its binary value is determined. If the binary value is greater than zero, the repeat indicator is set ON; otherwise, the repeat indicator is set OFF.

If the binary value of the repeat counter is greater than zero, the processor terminates the REPEAT command and gives control to the command to be repeated (the next command in sequence). If the binary value of the repeat counter is equal to zero, the processor accesses the first byte of the next command in sequence (the command to be skipped) and places that byte into the Q register. Bit 8 of the command code is examined to determine the length of the command being skipped (4 bytes or 8 bytes). The processor then increments the sequence control register (CR) according to the length of the command being skipped and terminates the REPEAT command; processing continues with the command addressed by the sequence control register.

- Using the Adder to Transfer Control

The hardware JUMP command and the eight hardware BRANCH commands enable the processor to transfer control to another point in the user's program (referenced by the effective A address); however, only the JUMP command provides the ability to return to the original program flow. The JUMP and BRANCH commands are normally 4-byte commands, containing only a command code (Q) and a 3-byte A operand; if a T character and B operand are included, they are setup in the appropriate registers for use by subsequent 4-byte commands, but are not used in the execution phase of these commands.

The eight BRANCH commands are: BRANCH EQUAL, BRANCH GREATER, BRANCH GREATER OR EQUAL, BRANCH LESS, BRANCH LESS OR EQUAL, BRANCH LESS OR GREATER, BRANCH OVERFLOW, and BRANCH UNCONDITIONAL. Each command, except the BRANCH UNCONDITIONAL command, tests the related hardware flags (less, equal, greater, or overflow) and, if the appropriate flag or flags are ON, transfers control to the command referenced by the effective A address in the LA register; the BRANCH UNCONDITIONAL command transfers control to the command referenced by the effective A address regardless of the hardware flags. The effective A address is established during command setup and is placed into the LA register before the processor enters the execution phase. During the execution phase, the processor tests the status of the hardware flags with relationship to the command code. If the appropriate flags are ON or the BRANCH UNCONDITIONAL command code is detected, the content of LA (effective A address) is placed into the sequence control register (CR), and control is given to the command referenced by that address. All BRANCH commands set the repeat indicator OFF (if ON) to prevent a repeat of the branch function.

The JUMP command unconditionally transfers control to the command referenced by the effective A address in the LA register; however, before the transfer is made, the processor saves the address of the next command in sequence, enabling the software to return to the normal program flow, if desired. During the execution phase, the processor places the content of CR (address of the next command in sequence) into the MB register where it is available for input to memory; it then forces a binary 34 (hex 0022) into the F terminal of the adder and places the output of the adder into the LC register, where it is used to address the link register (IR8). When IR8 has been accessed, the processor places the content of MB (address to be saved) into the link register, thereby providing a link to the normal program flow. Before terminating the command, the processor places the content of LA (effective address of the command to receive control) into the sequence control register and sets the repeat indicator OFF (if ON). Control is then given to the command addressed by the sequence control register.

- Using the Adder to Test Data

Three TEST commands (TEST BIT, TEST CHARACTER EQUAL, and TEST CHARACTER UNEQUAL) enable the processor to compare one byte of memory to the binary configuration of the command T character. Depending upon the result of the comparison, the processor transfers program control either to the command addressed by the effective A address or to the next command in sequence. The byte of memory being compared is referenced by the effective B address of the TEST command; if this address is computed using incremental addressing (Mode 3), the T character of the command can be compared to successive bytes in a field or to specific keys in a table by repeating the TEST command.

The TEST BIT command compares the individual bits of the T character to corresponding bits of the B-field character. If the B-field character has a corresponding 1 bit for every 1 bit of the T character, or if all bits of the T character are zero, the comparison is considered equal, the repeat indicator is set OFF (if ON), and program control is transferred to the command indicated by the effective A address. (The B-field character may have more 1 bits than the T character and still be considered equal.) If the comparison is not equal, control is either transferred to the next command in sequence or the TEST BIT command is repeated, depending upon the status of the repeat indicator.

The TEST CHARACTER EQUAL command compares the 8-bit binary configuration of the T character to the 8-bit binary configuration of the B-field character. If the comparison is equal, the repeat indicator is set OFF (if ON), and program control is transferred to the command indicated by the effective A address. If the comparison is unequal, control is either transferred to the next command in sequence or the TEST CHARACTER EQUAL command is repeated. The TEST CHARACTER UNEQUAL command works the same as TEST CHARACTER EQUAL, except that the transfer of control priority is reversed. If the comparison is unequal, the repeat indicator is set OFF (if ON), and program control is transferred to the command indicated by the effective A address; if the comparison is equal, control is either transferred to the next command in sequence or the TEST CHARACTER UNEQUAL command is repeated.

After command setup, the Q register contains the command code, the LA and LB registers each contain their respective effective addresses, and the T register contains the T character.

During the execution phase, the T register content is input to the G terminal of the adder, the output of the adder is complemented (all of the 1 bits are set to 0 and all of the 0 bits are set to 1), and the resulting configuration is stored in the MAP register. The B field is accessed according to the content of LB and is input to the G terminal of the adder, while the content of MAP is input to the F terminal. The complement of the T character is then binarily added to the B-field character, and the output of the adder (J01-J08) is tested for 1 bits. All output bits are equal to zero when the comparison is equal; any output bits not equal to zero indicate an unequal comparison for that bit.

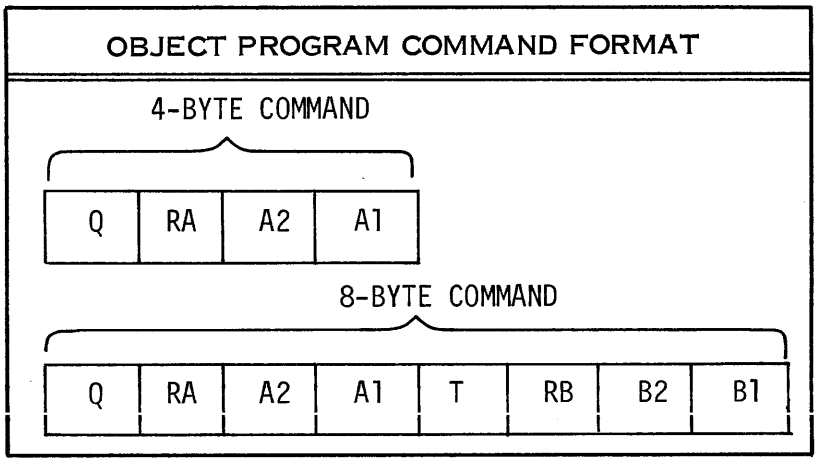
A control flag is set according to the output of the adder and the command code. This flag determines whether the content of LA (effective A address) is to be placed into the sequence control register (CR). If the control flag is ON, (indicating a successful comparison), the content of LA is placed into CR and control is transferred to that command. If the control flag is OFF, the sequence control register is unchanged, and the processor either transfers control to the next command in sequence or enters the repeat trapping flow.

FUNCTIONAL OPERATION OF THE ENTIRE ALU SECTION

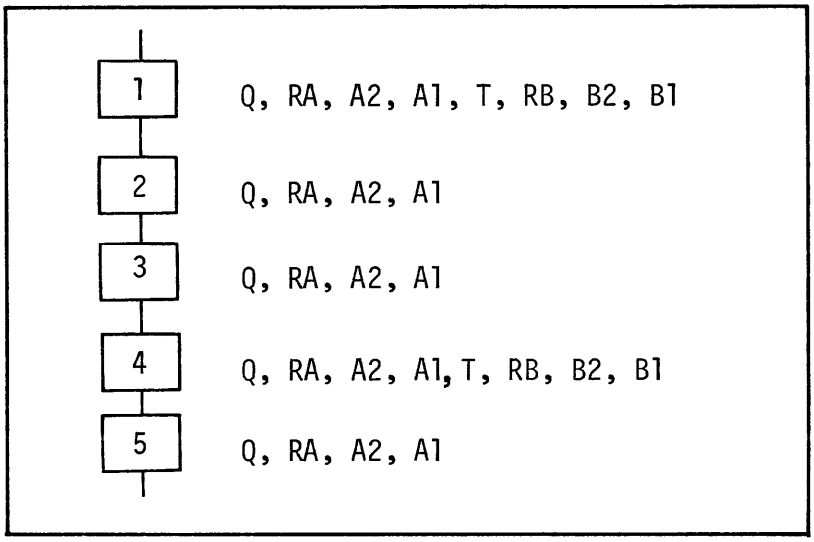
The functional description of an internal operation consists of five major parts: command format, command setup, command execution, between commands testing, and trapping flows. All definitions and explanations in the description assume that a program has been compiled and is resident in memory in either a 4-byte or 8-byte hardware-recognized format.

Command Format

Object program commands are stored in sequential order with one of two formats: (1) a 4-byte format, consisting of the command code (Q) and a 3-byte A operand, or (2) an 8-byte format, consisting of the command code (Q), a 3-byte A operand, a 3-byte B operand, and a T value.



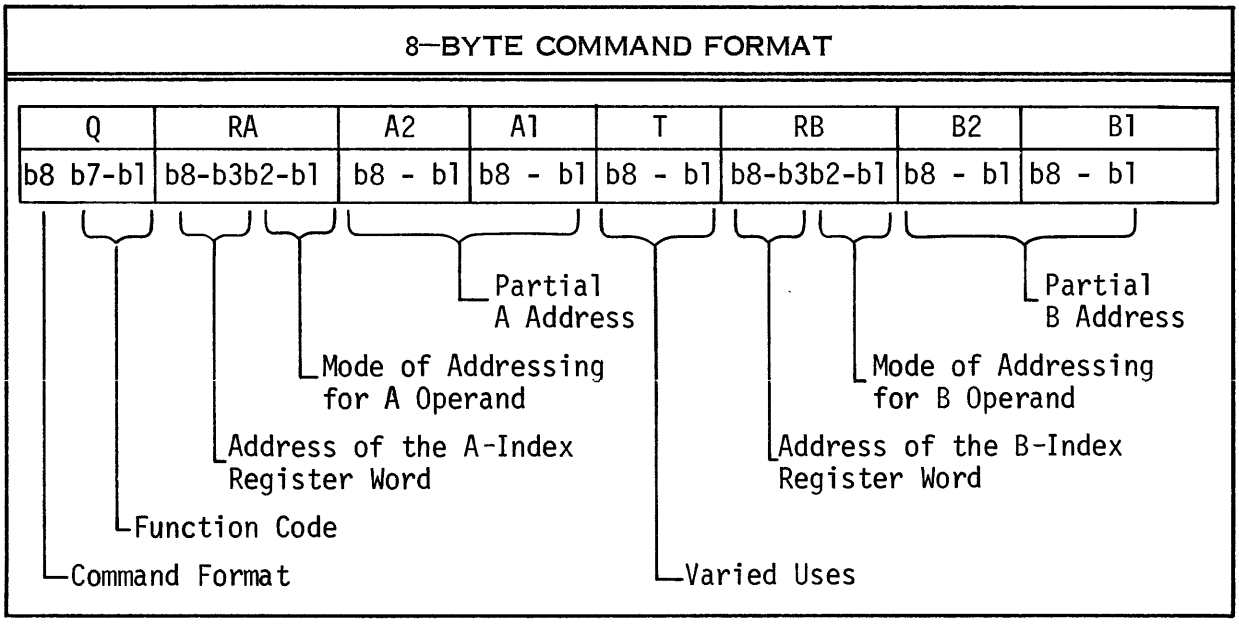
The processor normally requires two operands for command execution; therefore, when a 4-byte command is encountered, the processor uses the implied T value and B operand from the previously executed 8-byte command. In the following illustration, commands 2 and 3 use the implied T value and B operand of command 1; command 5 uses the implied T value and B operand of command 4. (The implied value is the value that exists in the hardware registers after command execution.)



The two command formats are functionally equivalent, since the processor saves the T value and B operand from the previously executed 8-byte command for use in successive 4-byte commands. Since the processor need not setup the T value and B operand each time, the 4-byte command saves a considerable amount of processor time.

The programmer must use caution when assigning 4-byte commands to ensure that the preceding 8-byte command contains the desired T and B values; in addition, he must take care that the execution of successive 4-byte commands does not change these values. Most commands restore the implied T and B values to their original values; however, some do not. (See the NCR Century 101 Hardware Command and Command Timing publication in this manual for individual commands.)

The following illustration and explanations describe the function of each byte in an 8-byte command; the first four bytes (Q, RA, A2, and A1) are identical in the 4-byte command.



- Command Code (Q)

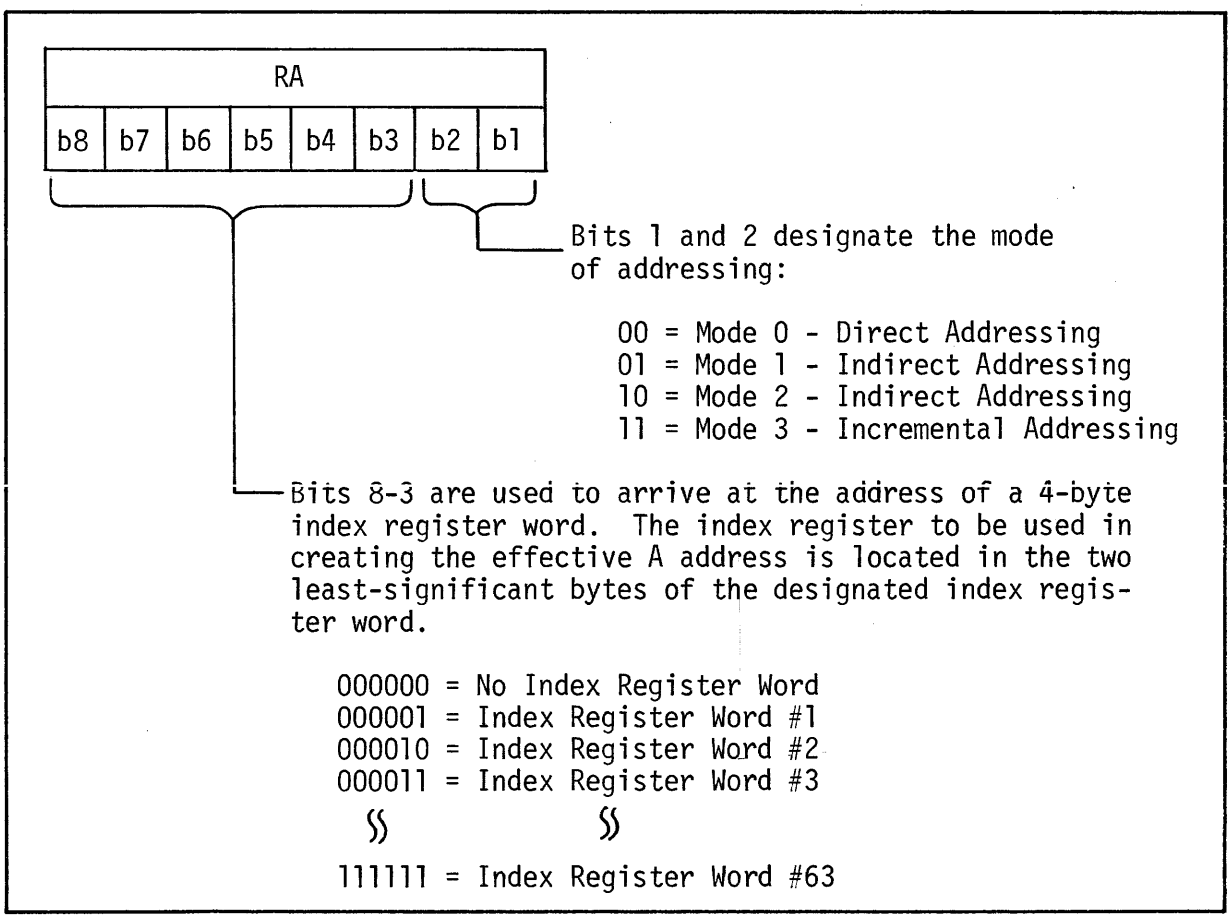
  The Q portion of the command specifies both the function to be performed and the format of the command. Thirty-seven hardware commands are available with the NCR Century 101 Processor, including three that are optional (MULTIPLY, DIVIDE, and LOGIC). All 37 commands are described in the NCR Century 101 Hardware Command and Command Timing publication in this manual.

  Bit 8 of the command code designates the format of the command, while bits 7-1 designate the function to be performed. For example, the ADD BINARY command may be stored in either of two ways: as a hexadecimal 60 (0110 0000) for an 8-byte command, or as hexadecimal E0 (1110 0000) for a 4-byte command. The only difference is bit 8; bit 8 is always OFF for an 8-byte command and ON for a 4-byte command.

- A Operand (RA, A2, and A1)

The A operand contains three bytes of information which are used to either arrive at the memory address for the A field, or to arrive at the address of another location in memory which may point to the desired A field.

A2A1 contains a 16-bit memory address which may be used as an effective A address or used in conjunction with RA to create an effective A address. RA performs two functions: it indicates the mode of addressing to be used, and it points to the index register (if any) to be used in creating the effective A address. (The four modes of addressing are explained under Special Addressing Modes in the Memory section of this publication.)



- T Value

The T value of the command has varied uses, depending upon the needs of the command; however, it is normally used to specify the length of the fields to be manipulated. The use of this character is discussed in detail with each command in the NCR Century 101 Hardware Command and Command Timing publication.

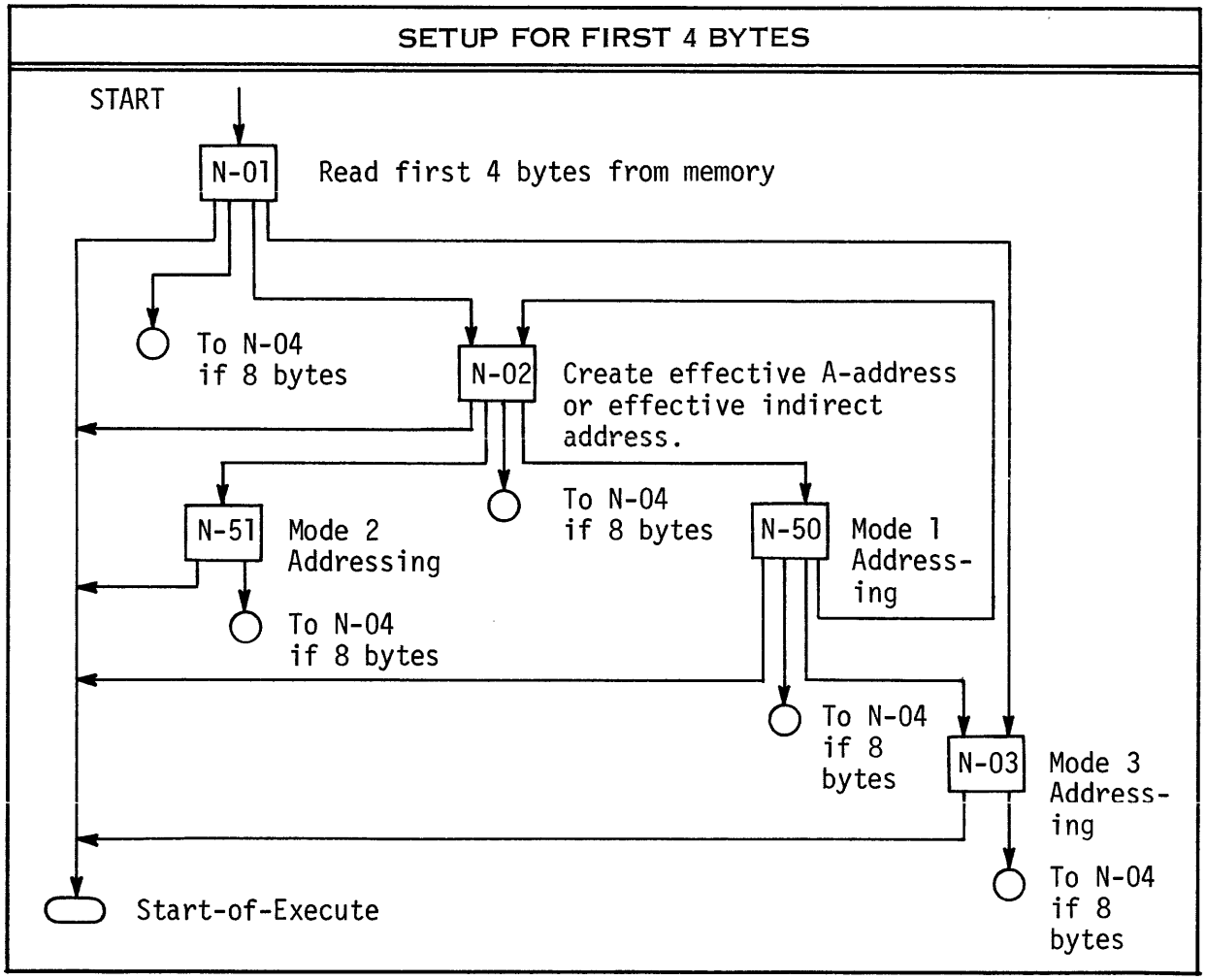
- B Operand (RB, B2, and B1)

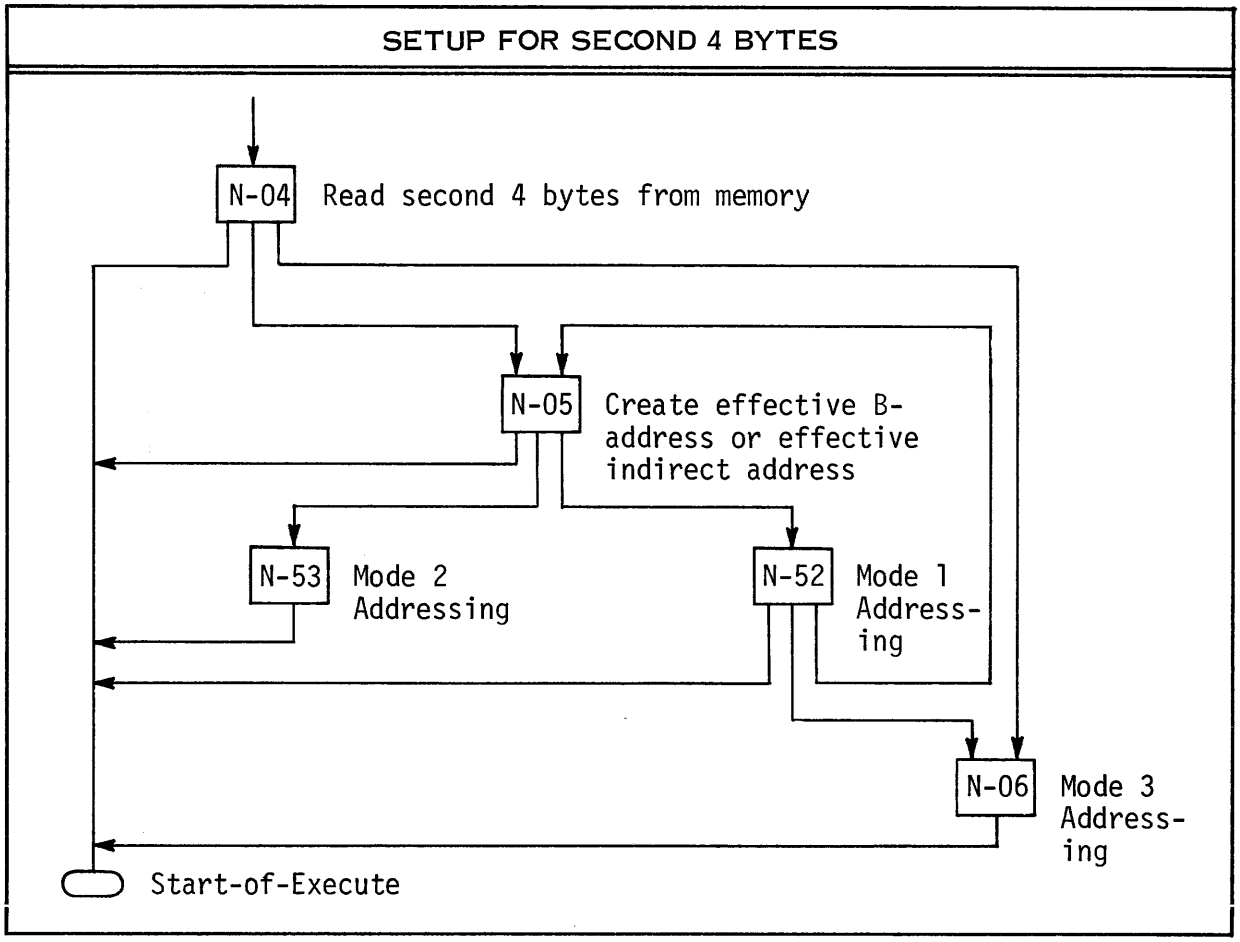
  The B operand contains three bytes of information which are used to either arrive at the memory address for the B field, or to arrive at the address of another location in memory which may point to the desired B field. The RB, B2, and B1 characters perform the same functions for the B operand as RA, A2, and A1 perform for the A operand.

Command Setup

The command setup phase reads the command from memory and places it in the appropriate hardware registers where it can be used to control the operation and to locate the fields to be manipulated. Command setup is performed by a series of logic flows which are designed to setup the first four bytes of the command (Q, RA, A2, and A1) and then, if necessary, the last four bytes (T, RB, B2, and B1) of the command.

There are 10 command-setup logic flows; each command requires the use of from 1 to 6 of these flows, depending upon the format of the command and the mode of addressing for the A and B operands. The following pair of illustrations indicate the relationship of one flow to another; a more detailed description of each flow is given in the explanation of the addressing modes.





All command setup functions begin at flow N-01, which performs the following steps:

- Accesses the first four bytes of the command to be executed (Q, RA, A2, and A1).
- Checks the command code (Q) for validity and format, then stores it in the Q register where it can be decoded for use during the execution phase.
- Places the partial A address (A2A1) into the LA register.
- Places the address of the associated index register (if any) into the LC register.
- Saves the previous T value and effective B address for use with a 4-byte command.
- Determines the addressing mode for the A operand.

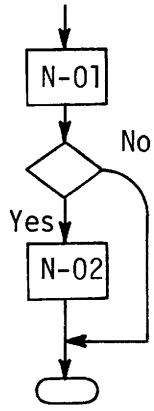
Command setup time is determined by the number of flows used, the command format (4-byte or 8-byte), and the mode of addressing for the A and B operands. The following examples indicate the setup time required for the A and B operands, depending upon the mode of addressing specified; total setup time is equal to the time required to setup both operands in an 8-byte command, or to the time required to setup the A operand in a 4-byte command.

- Setup Flows for the A Operand

  Setup time for the A operand depends upon the mode of addressing used.

- Mode 0 - Direct Addressing (A Operand)

Mode 0 addressing for the A operand uses only flow N-01 when no index register is specified; if an index register is specified, flows N-01 and N-02 are used. Total setup time for the A operand is either three or five cycles, depending upon whether an index register is specified.



(3-Cycles) Performs the normal N-01 flow functions.

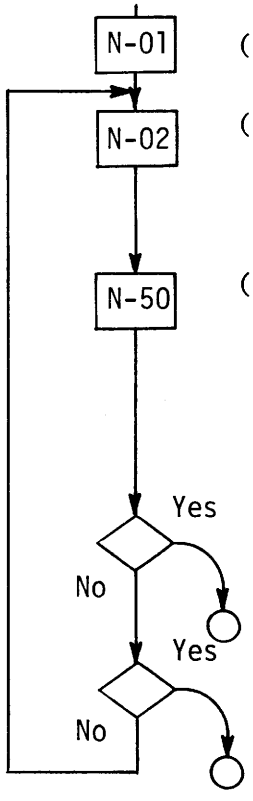
Did RA specify an index register? (If no, LA contains the effective A address.)

(2-Cycles) Add the memory contents of the associated index register to the contents of LA, creating an effective A address in LA.

Setup is completed for a 4-byte command; for an 8-byte command, go to flow N-04 to setup the B operand.

- Mode 1 - Indirect Addressing (A Operand)

Mode 1 addressing for the A operand uses flows N-01, N-02, and N-50. Total setup time for the A operand depends upon the number of passes through Mode 1, plus the number of cycles necessary for the final addressing mode (0, 2, or 3). The first pass through Mode 1 requires eight processor cycles, each additional pass requires five more processor cycles.



(3-Cycles) Performs the normal N-01 flow functions.

(2-Cycles) Adds the memory contents of the associated index register (if any) to the contents of LA, creating an effective indirect address in LA which points to another command (or 4-byte area) in memory.

(3-Cycles) Accesses the first 4 bytes of the new command; ignores the Q code, but places the address of the new index register (if any) into LC, places the new A2A1 address into LA, and determines the new addressing mode. Creates a tally to count the number of passes through N-50 (only 5 passes are permitted; a PE is indicated on the 6th pass).

Is this pass number 6?

PE error.

Did the new RA specify a change to Mode 0, 2, or 3?

Go to flow N-02 of the appropriate addressing mode to complete the setup.

- Mode 2 - Indirect Addressing (A Operand)

Mode 2 addressing for the A operand uses flows N-01, N-02, and N-51. Total setup time for the A operand is nine processor cycles.

N-01 (3-Cycles) Performs the normal N-01 flow functions.

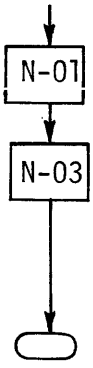
N-02 (2-Cycles) Adds the memory contents of the associated index register (if any) to the contents of LA, creating an effective indirect address in LA which points to a 3-byte area in memory that contains the desired A address.

N-51 (4-Cycles) Accesses the 3-byte area in memory and places the contents into the LA register, creating an effective A address.

Setup is completed for a 4-byte command; for an 8-byte command, go to flow N-04 to setup the B operand.

- Mode 3 - Incremental Addressing (A Operand)

Mode 3 addressing for the A operand uses flows N-01 and N-03. Total setup time for the A operand is six processor cycles.



N-01 (3-Cycles) Performs the normal N-01 flow functions.

N-03 (3-Cycles) Adds the memory contents of the associated index register to the contents of the LA register, creating an effective A address. Increments the contents of the associated index register by placing the effective A address into it.

Setup is completed for a 4-byte command; for an 8-byte command, go to flow N-04 to setup the B operand.

- Setup Flows for the B Operand

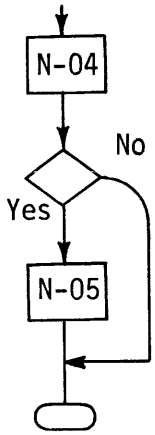
Upon completion of the setup phase for the A operand, the processor advances to flow N-04 to begin setup for the B operand. Flow N-04 performs the following functions:

- Accesses the next 4 bytes of the command to be executed (T, RB, B2, and B1).
- Places the T character into the T register.
- Places the partial B address (B2B1) into the LB register.
- Places the address of the associated index register (if any) into the LC register.
- Determines the addressing mode desired for the B operand.

Total setup time for the B operand depends upon the mode of addressing used.

- Mode 0 - Direct Addressing (B Operand)

Mode 0 addressing for the B operand uses only flow N-04 when no index register is specified; if an index register is specified, flows N-04 and N-05 are used. Total setup time for the B operand is either three or five cycles, depending upon whether an index register is specified.



(3-Cycles) Performs the normal N-04 flow functions.

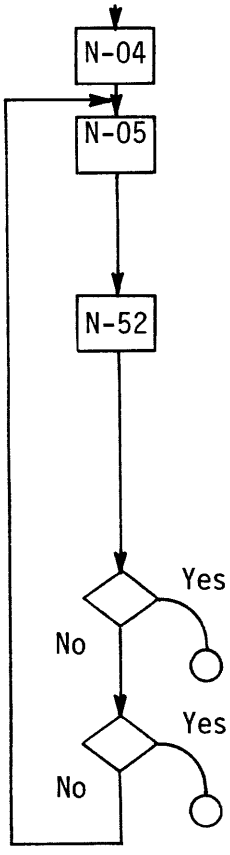
Did RB specify an index register? (If no, LB contains the effective B address.)

(2-Cycles) Adds the memory contents of the associated index register to LB, creating an effective B address in LB.

Setup is completed for an 8-byte command.

- Mode 1 - Indirect Addressing (B Operand)

Mode 1 addressing for the B operand uses flows N-04, N-05, and N-52. Total setup time for the B operand depends upon the number of passes through Mode 1 plus the number of cycles necessary for the final addressing mode (0, 2, or 3). The first pass through Mode 1 requires eight processor cycles, each additional pass requires five more processor cycles.



(3-Cycles) Performs the normal N-04 flow functions.

(2-Cycles) Adds the memory contents of the associated index register (if any) to the contents of LB, creating an effective indirect address which points to another command (or 4-byte area) in memory.

(3-Cycles) Accesses the last 4 bytes of the new command; ignores the T code, but places the address of the new index register (if any) into LC, places the new B2B1 address into LB, and determines the new addressing mode. Creates a tally to count the number of passes through N-52 (only 5 passes are permitted; a PE is indicated on the 6th pass).

Is this pass number 6?

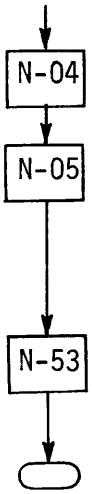
PE error.

Did the new RB specify a change to Mode 0, 2, or 3?

Go to N-05 of the appropriate addressing mode to complete the setup.

- Mode 2 - Indirect Addressing (B Operand)

Mode 2 addressing for the B operand uses flows N-04, N-05, and N-53. Total setup time for the B operand is nine processor cycles.



(3-Cycles) Performs the normal N-04 flow functions.

(2-Cycles) Adds the memory contents of the associated index register (if any) to the contents of LB, creating an effective indirect address in LB which points to a 3-byte area in memory that contains the desired B address.

(4-Cycles) Accesses the 3-byte area in memory and places the contents into the LB register, creating an effective B address.

Setup is completed for an 8-byte command.

- Mode 3 - Incremental Addressing (B Operand)

Mode 3 addressing for the B operand uses flows N-04 and N-06. Total setup time for the B operand is six processor cycles.

(3-Cycles) Performs the normal N-04 flow functions.

(3-Cycles) Adds the memory contents of the associated index register to the contents of the LB register, creating an effective B address. Increments the contents of the associated index register by placing the effective B address into it.

Setup is completed for an 8-byte command.

Command Execution

When command setup has been completed, the processor advances to the execute-logic flows. Selection of the individual N flows is dependent upon the interpretation of the Q character. The actual steps involved in executing a given command are subject to too many variables to permit a detailed explanation in this publication; each command is explained in the NCR Century 101 Hardware Command and Command Timing publication in this manual.

When command execution has been completed, the processor advances to a Between Commands Testing phase to determine whether any errors or interrupt conditions were detected.

System Malfunctions

System malfunctions are categorized as either command malfunctions or latent errors. Command malfunctions (memory errors, program errors, and illegal command codes) are detected during internal processing or during Between Commands Testing. Latent errors (program errors, memory errors, or transmission errors) are detected during I/O functions by either the I/O control logic or the printer control.

- Command Malfunctions

  As each command terminates, the processor performs the between commands testing function to determine the result of the operation. If a malfunction is detected (ME, PE, or ICC indicators ON), the processor enters the appropriate trapping flow, the malfunction indicator is set OFF, and the error indicator (EI) is set ON (EI is not set ON by the ICC trapping flow). The trapping flow preserves the status of the processor at the time of the malfunction and points to the appropriate software recovery routine. If another malfunction is detected before the processor enters the recovery routine, the processor enters an error halt state; otherwise, control is given to the first command in the routine and the error indicator (EI) is set OFF by the JUMP command in the routine. Between commands testing is performed for each command in the routine, with any malfunction resulting in an operator wait message.

- Latent Errors

  When a latent error is detected, the I/O control logic terminates the I/O function for the peripheral involved, sets the appropriate hardware indicator (ME or PE) ON, and inhibits the normal peripheral S3 status character. (An S4 status character is stored in the peripheral control word instead of the normal S3 status character; see "Processor Error Termination" under the I/O Control section of this publication.)

  The Between Commands Testing logic determines the trapping flow to be entered. If a latent ME or PE is indicated (either the ME or PE indicator is ON), the processor enters the appropriate trapping flow; if a latent transmission error is encountered and the IP indicator is ON, the processor enters the interrupt trapping flow. (If the IP indicator is not ON, the S4 status character indicates to the processor that the last transmission to the peripheral was unsuccessful and the software should initiate a retry.)

Between Commands Testing

Between Commands Testing (BCT) is a term given to a series of tests made by the processor upon termination of each command. The result of this testing determines whether the processor is to continue the logic flow, enter one of the trapping flows, or halt processing.

Between Commands Testing is actually a sequential check on various hardware indicators. If no error or interrupt conditions are indicated, the processor returns to the N-01 command setup flow to begin setting up the next command. If an error or interrupt condition is indicated, the processor enters an appropriate trapping flow.

The following items indicate the sequence in which the indicators are tested and specifies the trapping flow to be entered.

- If EI is ON together with ME, PE, ICC, or RE, the processor enters the error halt state.
- If EI is OFF and ME is ON, the processor enters the ME trapping flow.
- If EI is OFF and PE is ON, the processor enters the PE trapping flow.
- If EI is OFF and ICC is ON, the processor enters the illegal command code trapping flow.
- If RI is ON, the processor enters the repeat flow.
- If II and IP are both ON, the processor enters the interrupt trapping flow.

Trapping Flows

There are five trapping flows for the NCR Century 101 Processor: ME, PE, Illegal Command Code, Program Interrupt, and Repeat. (Each flow is actually part of one major trapping flow; however, for simplicity, they are explained separately.) Each trapping flow (except the repeat flow) stores the status of the processor at the time of error or interrupt, then transfers control to a software routine designed to determine the nature of the condition and to attempt a recovery whenever possible. The repeat flow simply alters the control register so that the following command can be setup and executed a predetermined number of times or skipped completely.

- ME Trapping Flow

  The ME trapping flow is entered whenever a parity error (even number of 1-bits or no 1-bits) is detected during a read function from memory, or during an addressing function for the read only memory (ROM).

  As the processor enters the trapping flow, it sets the EI (error indicator) ON and the ME indicator OFF. The status of the processor is then stored in index register words 5, 6, and 7 (hex location 0014-001F) as indicated by the following illustration. The OF, RI, G, E, and L bits of the error status word are normally set according to the status of their respective hardware flags and indicators; however, the RI bit may be set ON even though the hardware repeat indicator is set OFF by a conditional compare or test command.

ERROR STATUS FORMAT

MEMORY LOCATIONS (DECIMAL)											
0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
Q	NOT USED	EFFECTIVE A ADDRESS		T	NOT USED	EFFECTIVE B ADDRESS		FLG & IND	NOT USED	ADDRESS OF NEXT COMMAND	

b8	b7	b6	b5	b4	b3	b2	b1
NOT USED	NOT USED	OF	RI	NOT USED	G	E	L

For compatibility, all bits which are not used should be set to zero.

Once the processor status has been stored in the error status word, the OF, RI, and IP indicators are set OFF, and the error code indicator at memory location hex 0024 is set to indicate either a memory error condition or a ROM error condition (00000000 represents a memory error, 00000100 represents a ROM error). Finally, the processor transfers the address of the first command in the error routine from the ME control area (hex location 0100-0103) to the hardware sequence control register (CR) and, if no additional errors are detected, gives control to that routine. If an additional error (ME, PE, ICC, or RE) is encountered during the ME trapping flow, the appropriate indicator is set ON and the processor enters an error halt state. (If the error condition was the result of an illegal address, the sequence control register contains the illegal address.)

The error indicator (EI) is set OFF by the JUMP command at the start of the error routine, and processing continues at command setup flow N-01.

- PE Trapping Flow

The PE trapping flow is entered whenever a programming error is detected during command setup or execution. For example, all program addresses are checked for legality before being used to access memory; an address that is either equal to or larger than memory size, or not evenly divisible by four, is an illegal address. The PE trapping flow is also entered if the repeat indicator is ON and the repeat counter is equal to zero. The hardware MULTIPLY and DIVIDE commands check the operands being used for validity (proper length, zero modulo four addresses, etc.) and, if they are invalid, the PE trapping flow is entered (see the hardware MULTIPLY and DIVIDE commands in the NCR Century 101 Hardware Command and Command Timing publication for more information).

As the processor enters the PE trapping flow, it sets the PE indicator OFF and the EI indicator ON. The status of the processor is stored in the error status word (index register words 5, 6, and 7), using the same format as illustrated in the ME trapping flow. Once the error status word is stored, the OF, RI, and IP indicators are set OFF, and the PE error code (00000001) is stored in index register word 9 (hex location 0024). Finally, the processor transfers the address of the first command in the PE error routine from the PE/ICC control area (hex location 0104-0107) to the sequence control register (CR) and, if no additional errors are detected, gives control to that routine. If an additional error is encountered during the PE trapping flow, the appropriate indicator is set ON and the processor enters an error halt state. (If the error condition was the result of an illegal address, the sequence control register contains the illegal address.)

The EI indicator is set OFF by the JUMP command at the start of the error routine, and processing continues at command setup flow N-01.

- ICC Trapping Flow

The ICC trapping flow is entered whenever the processor encounters an invalid command code or an interpretive command. (An invalid command code is either one that is not contained in the hardware command code set, or one that cannot be recognized by the software as an interpretive.) In either case, the illegal command code trapping flow directs the processor to a

software routine which determines the command status (interpretive or invalid); and, if it is an interpretive, it contains the necessary hardware commands to perform the desired function.

As the processor enters the illegal command code trapping flow, the status of the processor is stored in the error status word (index register words 5, 6, and 7), using the same format as illustrated in the ME trapping flow. Once the error status word is stored, the ICC, OF, RI, and IP indicators are set OFF, and the ICC code (00000010) is stored in index register word 9 (hex location 0024). (The EI indicator is not set ON, unless another error condition such as PE is detected with ICC.) Finally, the processor transfers the address of the first command in the ICC routine from the PE/ICC control area (hex location 0104-0107) to the hardware control register (CR) and gives control to that routine. Processing continues as normal, beginning at command setup flow N-01 for the first command in the ICC routine.

- Interrupt Trapping Flow

The interrupt trapping flow is entered whenever the Between Commands Testing logic determines that the IP and II indicators are both ON. The IP (interrupt permit) indicator is set ON by program control, using the SET IP ON command; II (interrupt indicator) is set ON whenever the peripheral I/O is terminated and either an S3 or S4 status character is stored in the control word. The function of the interrupt trapping flow is to interrupt the normal processing flow, store the status of the processor at the time of interrupt, then transfer control to an interrupt routine which performs the necessary I/O function.

As the processor enters the interrupt trapping flow, it sets the IP and II indicators OFF and stores the status of the processor in index register words 10, 11, and 12 (hex location 0028-0033). The interrupt status is indicated in the following illustration:

INTERRUPT STATUS FORMAT

MEMORY LOCATIONS (DECIMAL)											
0040	0041	0042	0043	0044	0045	0046	0047	0048	0049	0050	0051
Q	NOT USED	EFFECTIVE A ADDRESS		T	NOT USED	EFFECTIVE B ADDRESS		FLGS & IND	NOT USED	ADDRESS OF NEXT COMMAND	

b8	b7	b6	b5	b4	b3	b2	b1
NOT USED	NOT USED	OF	RI	NOT USED	G	E	L

For compatibility, all bits which are not used should be set to zero.

Once the processor status has been stored in the interrupt status word, the OF and RI indicators are set OFF. The processor then transfers the address of the first command in the interrupt routine from the interrupt control area (hex location 0110-0113) to the hardware control register (CR) and gives control to that routine.

- Repeat Flow

When the REPEAT command is executed, the processor transfers the contents of the A field to the repeat counter at memory location 0020 (hex), where it can be tested for zero (zero indicates that the next command in sequence is to be skipped rather than repeated). If the repeat counter contains a value other than zero, the repeat indicator (RI) is set ON and control is given to the next command in sequence. The command to be repeated is then setup and executed as normal; however, after execution, the processor performs the Between Commands Testing test and finds the repeat indicator ON, causing the processor to enter the repeat flow.

As the processor enters the repeat flow, the repeat counter is tested for zero; if it is equal to zero, an immediate PE results and the flow is terminated. (RI remains ON and the repeat counter remains at zero.)

If the repeat counter is not equal to zero at the start of the flow, it is decremented by one and stored back into memory where it is again tested for zero; if it is still not equal to zero, the sequence control register is decremented by the size of the command being repeated (4 or 8 bytes), and the flow is terminated. Processing continues with the command being setup and executed again. If the repeat counter is decremented to zero, the repeat indicator is set OFF, the repeat flow is terminated, and control is given to the command addressed by the sequence control register (the next command).

NOTE

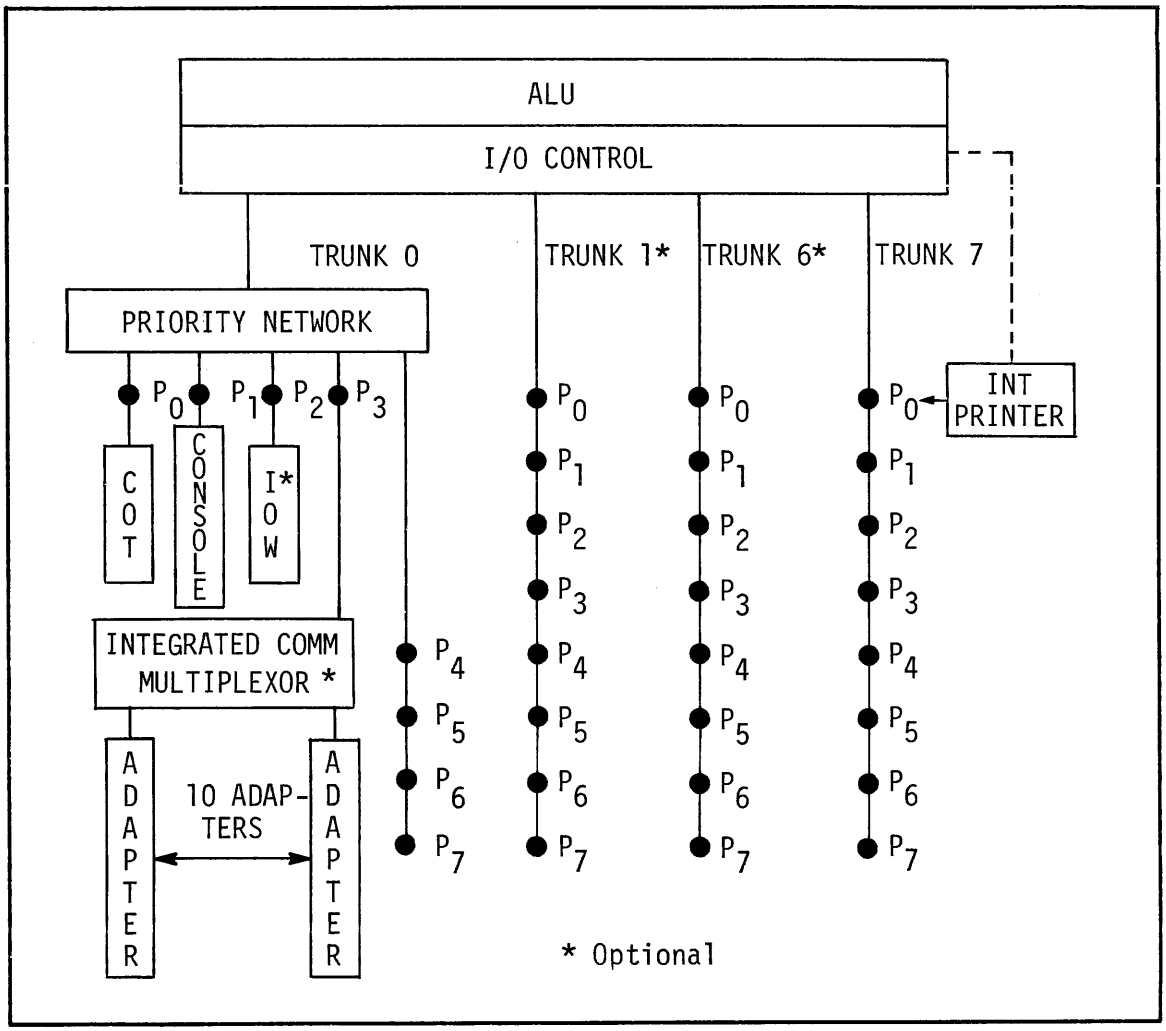
Certain commands set the repeat indicator OFF during execution and, therefore, cannot be repeated. See the repeat indicator explanation under Hardware Flags and Indicators in this publication for a list of commands and conditions that set RI OFF.

I/O CONTROL

GENERAL INFORMATION

The NCR Century 101 Processor uses the common trunk concept of I/O control, thereby separating the peripheral control lines from the data control lines. This separation enables the processor's arithmetic logic unit (ALU) to select the desired peripheral and initiate the I/O function, then return to the normal processing flow. When the peripheral is ready to send or receive a block of data, the I/O control logic interrupts the normal processing flow and transfers the data over the data control lines.

Four common trunks are available with the NCR Century 101 System, two are standard and the other two optional. Each common trunk has eight positions to which peripherals may be assigned; some positions are dedicated to specific integrated peripherals. The following illustration shows all four common trunks and indicates the dedicated trunk positions by showing the appropriate peripheral for each position. The COT and the operator's console are standard equipment with every processor; all other peripherals are optional. The dotted line connecting the integrated printer with the I/O control represents separate data transfer lines for the printer.



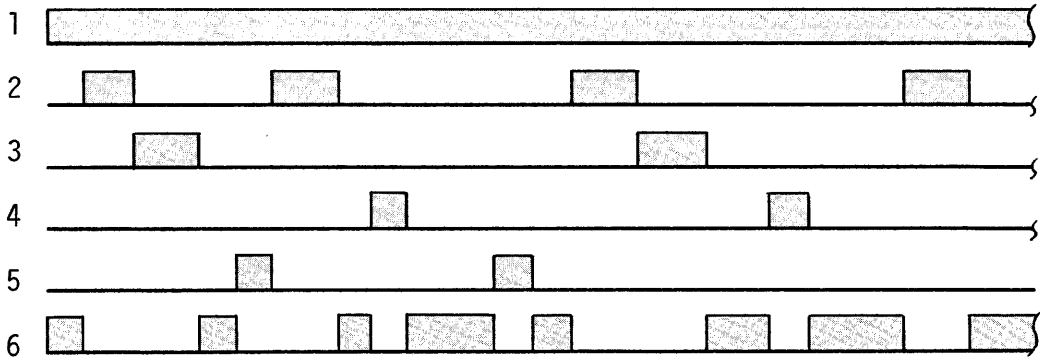
Trunk 0 has a bandwidth of 120,000 bytes per second, and trunk 1 has a bandwidth of 166,000 bytes per second; both of these trunks are classified as low-speed trunks. Trunk 6 with a bandwidth of 276,000 bytes per second and trunk 7 with a bandwidth of 416,000 bytes per second are classified as high-speed trunks. Trunks 0 and 7 are standard for all NCR Century 101 Systems; trunks 1 and 6, which are optional, may be included initially or added to the system when the need arises.

The common trunk concept of I/O control enables the processor to initiate up to five simultaneous I/O functions on trunk 0; for example, the peripherals at positions 0, 1, 2, and 3 may be operating simultaneously with another peripheral at position 4, 5, 6, or 7. The integrated printer (trunk 7, position 0) may also operate simultaneously with another peripheral on trunk 7. The basic system with trunks 0 and 7 may, therefore, initiate up to seven simultaneous I/O functions.

Since peripheral servicing does not normally require all of the memory cycles within a given time frame, the ALU "steals" the unused cycles and continues processing. If the ALU and I/O control both request a memory cycle at the same time, the I/O control is given priority.

The following illustration does not represent any particular system configuration or application, but rather illustrates the meaning of simultaneity. Assume that for a given time period, the ALU has initiated data transfer operations for all four common trunks and has returned to the normal processing flow. The I/O control interrupts the processing flow to steal back the necessary memory cycles for data transfer. There is never any simultaneous use of memory; however, four peripherals are simultaneously active with internal processing functions.

5—WAY SIMULTANEITY



1 Represents the continuously available memory time.

2-5 Represents the memory time taken by peripherals operating on trunks 0, 1, 6, and 7 respectively.

6 Represents the time available for internal processing between data transfer cycles.

If the I/O demand for memory time exceeds the memory time available, the system is considered overloaded. System overloads can usually be corrected by staggering the I/O requests (often accomplished by the system software); however, in some cases it may be necessary to alter the system configuration.

PERIPHERAL TYPES

There are two general classes of peripherals for an NCR Century System: integrated devices, which share some electronics and power supplies with the processor, and freestanding devices.

Integrated Peripherals

All integrated peripherals are assigned specific common trunk positions because of their need for additional logic and power supply. The following table lists the integrated peripherals available for use with the NCR Century 101 System and indicates their common trunk position.

PERIPHERALS	TRUNK	POSITION
Punched Card or Tape Reader	0	0
Operator's Console	0	1
Standard or Thermal I/O Writer	0	2
Communications Interface	0	3
Integrated Printer	7	0

All systems come with either a punched card or punched tape reader (COT) and an operator's console; the remaining peripherals are optional.

Freestanding Peripherals

Freestanding peripherals are divided into two classifications, level 1 and level 2.

- Level 1

  A level 1 peripheral contains the necessary circuitry to interface with the common trunk and has its own control logic for processor communication. These units may be attached to any common trunk position that is not reserved for an integrated peripheral with the following exceptions: the transfer rate of the peripheral cannot exceed the bandwidth of the trunk, and devices which have more than one control word active at a time must be placed on a low-speed trunk (these devices include the 621 Communication Multiplexors, the 622-601 Processor-to-Processor Intercoupler, the 622-301 OCR Control Unit, and the 626-101 Printer Controller with trunk grabbing).

- Level 2

  A level 2 peripheral requires a control unit or multiplexor to permit interfacing with the common trunk. The processor's I/O control communicates with the control unit or multiplexor, which in turn supervises the actual peripheral operation by means of its own trunk similar to the common trunk.

  A control unit or multiplexor may service one or more level 2 peripherals.

PERIPHERAL TRANSFER RATES

Peripheral transfer rate is an industry-accepted term that refers to the rate of speed (given in bytes- or characters-per-second) at which a peripheral or controller inputs or outputs data. This rate is a theoretical value based on continuous operation at full speed.

The peripherals commonly used with the NCR Century 101, and their transfer rates, are listed in the "Peripheral Transfer Rates and System Configurations" publication in the System Installation section of this binder.

TRUNK BANDWIDTH

Trunk bandwidth refers to the maximum rate of speed (given in bytes per second) at which a trunk may operate during an input or output operation. This rate is a theoretical value and may never be achieved because of a lower transfer rate for the peripheral or controller being serviced.

The following table indicates the maximum bandwidth of each trunk on the NCR Century 101 System.

TRUNK	BANDWIDTH
0	120 KB*
1	166 KB
6	276 KB
6 (buffered)	833 KB
7	416 KB

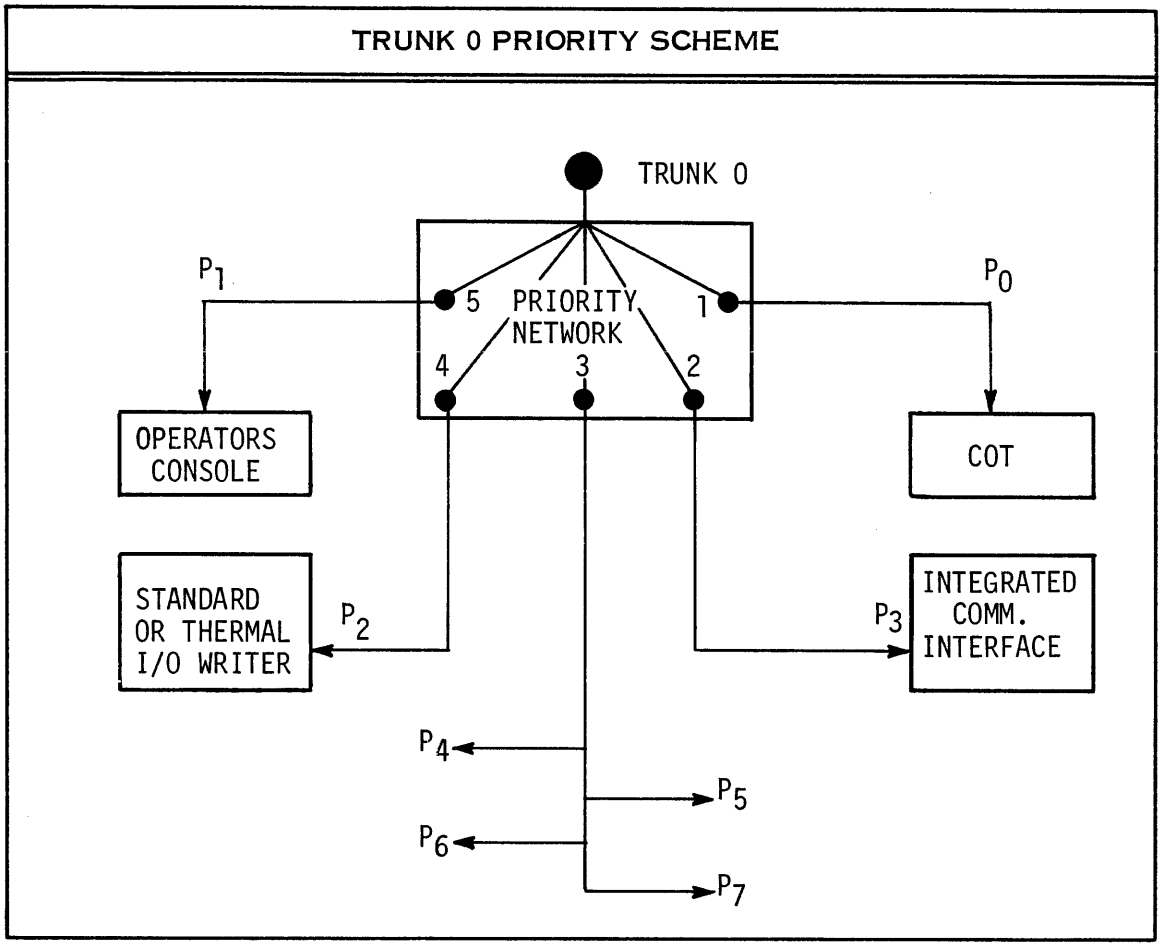
*The total transfer rate of all peripherals operating simultaneously on trunk 0 must not exceed 166 KB; the maximum transfer rate of any individual peripheral on trunk 0 must not exceed 120 KB.

TRUNK AND PERIPHERAL PRIORITY

The I/O control logic determines the priority for peripheral input or output, based on the bandwidth of the trunk and the speed of the peripheral. Usually, the trunk priority is 7 (highest), 6, 1, 0; however, if the integrated printer or the COT are not serviced within a specified time, they are given first priority. In addition to the I/O control priority scheme, trunk 0 has a hardware-wired priority network which establishes a separate priority scheme for its eight positions (this network is effective only after trunk 0 is given priority by the I/O control logic).

Trunk 0 Priority Network

The following illustration shows the 5-position priority network for trunk 0. Each position indicates (by its number) the priority of the trunk position to which it is assigned.



Position 1 of the network, which selects the COT at P_0 of the trunk, has highest priority; position 2 of the network has second priority, etc. If positions 2 and 3 of the network request service at the same time, position 2 (communications interface) receives first priority and position 3 second priority. On the other hand, if position 1 of the network requests service while position 2 is being serviced and before position 3 is serviced, position 1 will be given priority ahead of position 3.

I/O Control Priority Scheme

The I/O control logic initially establishes priority according to the bandwidth of the trunk; trunk 7, therefore, has the highest priority and trunk 0 the lowest priority. If these trunks are not requesting service, the I/O control logic gives priority to the integrated printer. Finally, if no I/O requests are received, control is given to the processor for internal processing.

FIRST FLOW PRIORITY ORDER					
1st	2nd	3rd	4th	5th	6th
T7	T6	T1	T0	INTEGRATED PRINTER	INTERNAL PROCESSING

The chart above illustrates the order of priority for the first I/O flow only; once an I/O flow is initiated on a given trunk, the priority scheme for succeeding I/O operations may change. For example, special consideration is given to the integrated printer and COT. If the integrated printer is not serviced within 7 microseconds after it requests service, it is given highest priority; if the COT is not serviced within 300 microseconds after it requests service, it is given priority after the printer. In addition, the high-speed trunks (trunks 6 and 7) are checked more often than the others, thereby giving them special attention.

SYSTEM I/O BANDWIDTH

System I/O bandwidth represents the maximum rate at which a given computer system can accommodate data transfer when all processor cycles are devoted to I/O handling. This rate is determined by the memory speed, the number of processor cycles necessary to transfer a byte of data, the characteristics of the peripherals involved, and the system priority scheme. The maximum bandwidth of the NCR Century 101 System is 833 KB; however, this rate is obtained only when using the double-buffered trunk 6. If I/O functions require the use of integrated peripherals or low-speed trunks (trunks 0 or 1), the system I/O bandwidth ranges between 120 KB and 833 KB.

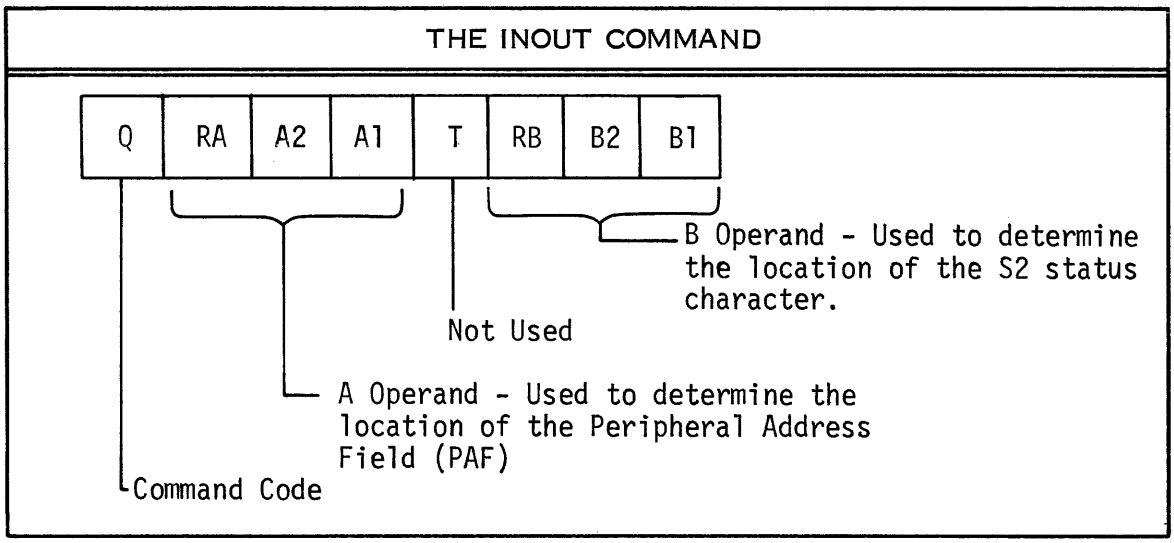
If the accumulative transfer rate of simultaneous I/O operations attempts to exceed the system I/O bandwidth, an overload condition occurs. Normally, the peripheral causing the overload deselects and stores an overload status character; some peripherals, however, are capable of operating at a lower transfer rate and do not deselect or store an overload status character. If an overload condition occurs, the processor continues functioning, and all other I/O operations currently in process continue as normal; the system software may either retry the operation causing the overload or display a message to the operator if manual intervention is required.

A reasonable number of system overloads represents efficient use of the hardware and I/O bandwidth by assuring that maximum use is being made of all available processor cycles; system overloads need not degrade system performance.

I/O FUNCTIONAL OPERATION

Each I/O operation is functionally divided into three parts: peripheral selection, data transfer, and termination of the I/O operation.

Peripheral selection is handled by the ALU, based on information derived from the INOUT command. This command provides the ALU with the location of the Peripheral Address Field (PAF) and the location in memory of the S2 status character, which indicates whether the selection was successfully completed.



Once the peripheral is selected, the ALU gives the I/O control logic the responsibility for data transfer and returns to the normal processing flow. I/O control logic has the capability to interrupt the processing flow to allow data transfer whenever the peripheral is ready to send or receive a block of data.

An I/O operation is normally terminated by the processor when the NA portion of the peripheral control word equals the TA portion; it may also be terminated by the peripheral when an End-of-Block condition is encountered. In addition, the processor and peripheral both contain the logic to detect a parity failure and to terminate the I/O operation if one is detected. The peripheral also terminates the I/O operation if the operator manually interrupts the operation by placing the peripheral in standby.

Each subject (peripheral selection, data transfer, and termination of the I/O operation) is discussed in more detail under individual headings.

Peripheral Selection

The ALU selects the desired peripheral and determines its function according to the contents of its Peripheral Address Field in memory. Each peripheral has its own PAF to indicate its location (trunk, position, and unit number) and the function it is to perform.

Since some peripherals require more information than others, the PAF is variable in length. For example, a card reader/punch may perform either a read or a punch function; therefore, it requires a function code to indicate the type of operation. A controller servicing one or more peripherals requires an additional selection code to select the proper peripheral, etc.

The following illustration shows only the first character of the PAF, the position character. This character is necessary for the selection of all peripherals because it contains the trunk and position number.

PAF POSITION CHARACTER							
b8	b7	b6	b5	b4	b3	b2	b1
0	T	T	T	0	P	P	P

TTT = I/O trunk number (0 through 7)
PPP = Position number (0 through 7)

Unused bits in the PAF character are usually zero for compatibility with other members of the NCR Century family.

The arrangement of other characters in the Peripheral Address Field is determined by the type of peripheral to which it applies:

- Level 1 devices that perform only one function require only the position character.
- Level 1 devices that perform more than one function, but require no further addressing, require a function code following the position character. (Function codes for a particular peripheral are discussed in the individual peripheral publications in this manual.) Function codes may be included for software purposes even though the device does not require one.
- Level 1 devices that serve as controllers for one or more level 2 devices require an additional selection character; that character may range in value from 0 through 255. (More remote levels of selection may require even more selection characters.)

The address of the first PAF character is obtained from the effective A address of the INOUT command. When the ALU encounters an INOUT command, it locates the desired PAF and sends the first character to the trunk and position indicated; the peripheral at that location then informs the processor of its status: ready, busy, in standby, or inoperative. One of the following steps are taken, depending upon the status of the peripheral:

- If either the trunk or the peripheral is busy, or if the peripheral is inoperative, the appropriate status character is stored in memory and the command terminates.

- If the trunk and peripheral are ready (not busy, not in standby, not inoperable), the remaining portion of the PAF is transmitted to the peripheral, one character at a time. When the peripheral signals that all necessary PAF characters have been received, the "command-initiated" status character is stored and the command terminates.

- If the required peripheral responses are not received for any reason (no unit, illegal function code, etc.), the "inoperative" status character is stored and the command terminates.

A status character is stored in memory at the location specified by the effective B address of the INOUT command. This code, which consists of one of the following bit configurations, is called the S2 status code and indicates the result of the selection attempt.

- 10000000 (BUSY)

  The busy status indicates that either the I/O trunk is tied up with another peripheral or that the peripheral itself is busy. (The control unit for CRAM or disc can suppress the busy status and permit a seek function even though another peripheral under the same controller is presently engaged in a seek, read, or write operation; to seek means to locate the track where information is to be read or written.)

- 10000010 (STANDBY)

  The standby status indicates that the peripheral has been put in a standby condition by the operator.

- 00000010 (INOPERATIVE)

  The inoperative status indicates either that the peripheral did not respond to the selection attempt or that it did not respond within the allotted time.

- 01000000 (COMMAND INITIATED)

  The command-initiated status indicates that the peripheral has accepted all PAF characters and selection has been completed.

Data Transfer

When peripheral selection is completed and the peripheral or controller is ready to send or receive data, it sends a response number to the processor. The response number, which may range from 0 through 255, is wired into the peripheral at installation time, but (except for integrated peripherals) does not necessarily relate to the trunk and position number of the peripheral.

The following response numbers are assigned to the integrated peripherals.

INTEGRATED PERIPHERAL RESPONSE NUMBERS	
PERIPHERAL	NUMBER
COT	00
OPERATOR'S CONSOLE	01
INTEGRATED PRINTER	02
TELETYPE OR THERMAL I/O WRITER	03

The I/O control logic uses the response number to calculate the address of the control word for the selected peripheral (each peripheral has an 8-byte control word which may be used to address information, to indicate when to terminate the operation, and to store the terminating status character).

There are 256 8-byte control words available for I/O use, beginning with control word 0 at memory location 1024 (decimal). To determine the control word address for a specific peripheral, the I/O control logic effectively multiplies the response number by 8 and adds the result to 1024.

All control words except the integrated printer control word contain the same type of information; the integrated printer control word, however, is specially designed to control line advance and printer control functions (see the reserved memory section of this publication or the integrated printer publications in this manual).

This publication considers only the standard peripheral control word for the discussion of data transfer.

STANDARD CONTROL WORD FORMAT

S	NA			TA		NOT USED
	N3	N2	N1	T2	T1	

S = A 1-byte field where the peripheral status (S3 or S4) is stored.

NA = A 3-byte field which contains the memory address of the next character to be transferred by I/O control. NA is incremented by 1 as each character is transferred. (N3 is not used on the NCR Century 101 and should be set to 0 for compatibility with other NCR Century Systems.)

TA = A 2-character field which contains the terminating address. TA is compared with the N2 and N1 characters of NA to determine when to terminate the I/O operation. When NA equals TA, data transfer is completed.

When the peripheral is ready to send or receive a character, the I/O control logic interrupts the normal processing flow (providing the processor Interrupt Permit indicator is ON) and steals the necessary memory cycles for data transfer (see the interrupt trapping flow in the ALU section of this publication). The I/O control logic then transfers the data character to or from memory according to the address specified in the NA portion of the control word. After the data transfer, the I/O control logic increments the NA character by 1 and stores it back into the control word.

If NA is not equal to TA before incrementation, the ALU returns to the normal processing flow again, leaving the peripheral under I/O control. When the peripheral is ready for the next transfer of data, the I/O control logic interrupts the normal processing flow, transfers the data, and increments the NA field. This process continues until NA equals TA or until some other condition terminates the I/O operation.

The NA-TA fields permit a maximum record length of 65,536 characters; however, the limit imposed on the system is determined by the capabilities of the peripheral and the processor's memory size.

Termination of the I/O Operation

I/O termination takes place when the I/O control logic determines that all of the required data has been transferred (NA = TA), when the peripheral encounters a terminating condition (End-of-Block), or when an error condition is detected by either the processor or the peripheral.

At termination, the peripheral sends an 8-bit status character (called S3 or S4, depending upon its origin) to the processor. The bit configuration of this character indicates the outcome of the I/O operation and is stored in the peripheral control word for use by the system software in determining the action to be taken.

There are four ways to terminate an I/O operation: normal processor termination (NA=TA), normal peripheral termination (End-of-Block), peripheral error termination, and processor error termination.

- Normal Processor Termination

  When NA equals TA, the processor sends a terminate signal to the peripheral; the action taken by the peripheral is determined by the type of operation being performed, the type of peripheral, and the mode of operation (real-time or offline).

If the peripheral is a realtime, level 1 device (such as a communications multiplexor) and is controlling a remote terminal with more data to transfer, it responds with the S3 Segment Complete status. The processor must then initiate a read or write function for the remote terminal before the next data character arrives at the control unit; failure to do so results in a program overload condition.

For offline operations or realtime operations with no more date to transmit, the peripheral responds with an appropriate S3 status character and deselects. If the peripheral is reading, it may continue to move the media until an End-of-Block (EOB) condition is encountered (some peripherals, such as the paper tape reader, do not detect EOB conditions and stop reading immediately). If the peripheral is writing, it initiates an EOB write operation immediately.

- Normal Peripheral Termination

If an End-of-Block condition (physical end of card, tape gap, special character, etc.) is encountered before a processor terminate signal is recieved, the peripheral requests service and sends a terminate signal to the processor, along with an appropriate S3 status character.

- Peripheral Error Termination

If the peripheral encounters any of the following conditions, it immediately sends a terminate signal to the processor, along with an appropriate S3 status character:

- Error when writing.
- System overload when writing.
- Write lockout when attempting to write, erase, or rewind (rewind is initiated, but write and erase functions are not).
- Peripheral inoperative when attempting to perform any function.
- Transmission errors detected by the peripheral.
- Special conditions such as attempting to write a record (on disc) which is longer than the sector length, or detection of a special character on magnetic tape when reading.

Other error conditions detected by the peripheral may not be transmitted to the I/O control until an EOB is detected or until the next I/O command is initiated. These conditions are:

- Error when reading (this status is sent to the processor when the EOB is encountered).
- System overload when reading (this status is sent to the processor when an EOB is encountered).
- Program overload (this status is sent to the processor on the next INOUT command).

- Processor Error Termination

  If the I/O control logic detects a transmission error during data transfer, it signals the peripheral to stop sending data, inhibits the S3 status character (replacing it with an S4 status character), and deselects the peripheral.

In all four termination cases, the status of the I/O operation is indicated by the bit configuration of the S3 or S4 status character, which is stored in the peripheral control word.

- S3 Status Characters

  The S3 status character indicates one of three situations: (1) the operation has been completed, (2) the transfer of a segment of data has been completed, or (3) the operation has been terminated early due to a transmission error or operator intervention.

  Bits 7 and 8 of the S3 status character indicate which of the three situations exists. If bits 7 and 8 are both OFF (00XXXXXX), the operation has been completed; if they are both ON (11XXXXXX), a segment of data has been transferred. If only bit 8 is ON (10XXXXXX), the operation terminated early. Bits 6 through 1 indicate the outcome of the operation. All six bits OFF indicate a successful operation; any other configuration indicates a particular type of condition.

  The following S3 definitions are general. Certain peripherals have specific status characters which are explained in separate publications dealing with those devices. It is possible for more than one S3 condition to occur on a given I/O operation; therefore, it is the peripheral's responsibility to relay a bit configuration indicative of all conditions by setting the appropriate bits to 1.

S3 STATUS CODES		
CODE	TITLE	MEANING
00XXXXXX	Operation Complete	The I/O operation has been completed.
11XXXXXX	Segment Complete	A segment of data transfer has been completed, but the realtime peripheral has more data to transmit to its controller.
XX100000	Error	The peripheral has detected an error (normally a parity error) during the I/O operation.
XX010000	System Overload	The I/O control did not answer the peripheral's request for service within its character time.
XX001000	Media	The peripheral has detected a warning marker (such as a magnetic tape warning marker) during a write operation.
XX000100	Write Lockout	Write lockout indicates that the peripheral attempted to write while in a lockout state.
XX000010	Inoperative	The peripheral is inoperative.
XX000001	Special	Covers all conditions not specified above.
10000001	Transmission Error	A transmission parity failure has been detected by the peripheral.
10000010	Standby	The peripheral is in a standby state.

- S4 Status Character

  The I/O control logic generates and stores an S4 status character in the peripheral control word when either the I/O control or printer control detects a latent ME, PE, or transmission error. The I/O control logic also causes the peripheral to stop transferring data, to inhibit sending an S3 status character, and to deselect.

There are three S4 status characters.

S4 STATUS CHARACTERS		
CODE	TITLE	MEANING
10000001	Latent Transmission Error	A transmission parity error has been detected by the I/O control.
10000100	Latent ME	A memory error has been detected by I/O control or the printer control.
10001000	Latent PE	A programming error has been detected by the I/O control or the printer control.

The following descriptions define the state of memory after the detection of a latent error.

- Latent Transmission Error

  If the error occurred on a low-speed trunk (0 or 1), NA usually contains the address of the character which caused the error; the only exception is when the error occurs on the S3 status character (then NA is equal to TA). On high-speed trunks (6 and 7), NA usually contains the address of the character which caused the error, plus one. Again, the exception is on an S3 status error (then NA is equal to the address of the last character transmitted).

- Latent ME

  In the case of a latent ME, the character causing the error is not transmitted. If the error is detected during data transfer on one of the low-speed trunks (0 or 1), NA contains the address of the character that caused the error; if the error occurs in the control word, the value of NA and TA is determined by the error. When a parity error is detected during data transfer on a high-speed trunk (6 or 7), NA normally contains the address of the character that caused the error, plus one; the exception to this is when the error occurs in the control word (then NA and TA are determined by the error).

- Latent PE

  If a programming error occurs on a low-speed trunk (0 or 1), the value of NA after the error depends on its initial value and the memory size; for example, if the initial value of NA is greater than memory size, it is left unchanged.

  On the other hand, if the initial value was less than memory size, but is incremented to equal memory size, NA is one greater than memory size after the error (for systems with 64K memory, NA contains FFFF after the error).

If a programming error occurs on a high-speed trunk (6 or 7), the value of NA after the error depends on its initial value, the memory size, and the peripheral transfer rate. When the initial value of NA is greater than memory size, NA remains unchanged after the error; however, if NA is incremented to equal memory size, its value after the error is determined by the transfer rate of the peripheral: greater than 120 KB leaves NA unchanged, 120 KB or less increments NA by 1.

OPTIONS

GENERAL INFORMATION

The basic NCR Century 101 Processor contains a 16K-byte core memory, two common trunks (trunks 0 and 7), an operator's console, 34 hardware commands, and the logic and electrical power necessary to operate the standard printer and COT (card or tape) reader. The following features are optional.

EXTENDED MEMORY

The processor's memory can be expanded in 8K increments from 16K bytes to 32K bytes, then in 16K increments to 64K bytes, then in 32K increments to 128K bytes. Your processor must be equipped with the Extended Addressing Logic (EAL) feature to provide the proper addressing scheme for memories larger than 64K bytes.

I/O TRUNKS 1 AND 6

Two additional common trunks can be added to the basic processor, thereby providing the system with additional I/O capabilities. Each optional trunk contains eight positions which may be used to service freestanding peripherals. The bandwidth of trunk 1 is 166 KB and the bandwidth of trunk 6 is 276 KB; the double-buffered trunk 6 has a bandwidth of 833 KB.

50-HERTZ ELECTRICAL CAPABILITIES

The processor normally operates on 60-Hertz single-phase or 60-Hertz three-phase electrical source; however, it may be equipped to operate on 50-Hertz single-phase or 50-Hertz three-phase electrical source if desired.

REMOTE ALARM

The processor may be equipped with either a remote bell or a remote audible alarm, either of which operates in parallel with the console's audible alarm.

INTEGRATED CARD OR TAPE READER (COT)

The basic system includes either the 682-101 Integrated Card Reader or the 662-100 Integrated Punched Tape Reader. The COT (card or tape) reader is located on the processor, next to the operator's console; its controls are built into the operator's console. Electrical power and control logic, both of which are provided by the processor, are available at trunk 0 position 0. The COT reader is selected by a 1-character Peripheral Address Field (PAF) that contains the trunk and position number. After selection, the COT reader uses peripheral control word 0 at memory location 1024 (decimal) to: (1) store the peripheral status character, (2) locate the next memory address to be accessed, and (3) determine when to terminate the I/O operation.

The following explanations describe the basic features of the integrated card and punched tape readers; however, a detailed explanation of each reader can be found in this manual under the PUNCHED MEDIA PERIPHERALS tab, "682-100 Integrated Card Reader" and "662-100 Integrated Punched Tape Reader."

682-101 Integrated Card Reader

The 682-101 Integrated Card Reader reads industry standard, 80-column cards (including round-cornered cards) at a maximum rate of 300 cards per minute. This reader has one input hopper and one output stacker, each with a capacity of 1000 cards. Cards are read one column at a time, beginning with column 1. As the card passes through the optical reading station, the photocells sense the punches in the card and generate a 12-bit Hollerith character which is converted to an 8-bit binary configuration by a hardware decode matrix; this 8-bit configuration is then sent to the processor, where it is converted to an 8-bit ASCII code by the system software.

662-100 Integrated Punched Tape Reader

The 662-100 Integrated Punched Tape Reader reads strips or rolls of tape, which may vary in length from a minimum of one foot (one character plus leader) to a maximum of 350 feet. The reading rate of the integrated tape reader is 1000 characters (100 inches) per second. Any industry-standard punched paper tape or paper-mylar tape with 5, 7, or 8 data channels may be used; in addition, the tape may be dry or oiled and may be of any color, providing that it does not exceed 70% transparency. The integrated tape reader reads tapes punched in any customer-defined code set with either even or odd parity control. The NCR Century Operating System, however, contains only thoses tables necessary, to translate the Hollerith Extended A or H code sets to NCR Century ASCII codes; any other code set must be defined by the programmer and compiled with the user's program.

PRINTERS

The basic system normally includes a 649-300 freestanding printer; however, depending on the site requirements, the system may contain a 640-102 or 640-132 integrated printer instead (the 640-132 is essentially the same as the 640-102, except that it has a print rate of 300 lines per minute; the 640-102 prints 450 lines per minute).

If an integrated printer is used, it must be assigned to trunk 7, position 0, which provides special control logic and electrical power for the printer. Although the integrated printer occupies a common trunk position, it operates independently from all other peripherals on that trunk; that is, special data transfer lines enable the printer to transfer data simultaneously with any other peripheral on trunk 7. The integrated printer is selected by a 2-character Peripheral Address Field (PAF). The first byte of the PAF selects the printer at trunk 7, position 0; the second designates the function to be performed (print only, line advance only, or line advance and print). After selection has been completed, the integrated printer uses peripheral control word 2 at memory location 1040 (decimal) to: (1) store the peripheral status character, (2) locate the beginning address of the field to be printed, (3) determine when to terminate the I/O operation, and (4) determine the number of lines to advance the paper.

The following explanations describe the basic features of the 649-300 free-standing printer and the 640-102 integrated printer. A detailed explanation of each printer can be found in this manual under the PRINTERS tab, "649-300

Printer" and "640-102 Printer". (Also avaialble for use with the NCR Century 101 processor are the 646 and 647 high-speed train printers, with print speeds --in burst mode-- of up to 3500 lines per minute. These printers are described in this manual under the PRINTERS tab, "The 646 and 647 Train Printers".

649-300 Freestanding Printer

The 649-300 freestanding printer is housed in a single cabinet, which contains all of the logic circuitry, printing mechanisms, and the power supply for the printer. It is a low-speed peripherals, having a maximum print speed of 300 lines per minute and a transfer rate of 9.1 KB, and is therefore suitable for use on any low-speed trunk. The 649-300 printer is equipped with trunk-grabbing capabilities, whereby it frees the trunk for use by another peripheral as soon as it initiates a printing or paper-feed operation, then grabs the trunk back to transmit its status character at the completion of that operation.

The print line consists of a revolving steel drum equivalent in length to 132 print columns, with 63 printable characters and a space in each printing column. However, every other column is blank. Thus, to print a line of 132 characters the 649-300 prints half the characters (either the odd-numbered columns or the even-numbered columns), shifts the paper one-tenth of an inch horizontally, and prints the other 66 characters. It does not shift the paper back again during the same operation. For example, if the paper is in the rightmost position (its "home" position), the printer first prints the odd-numbered columns, shifts the paper to the left, then prints the even-numbered columns; on the next operation, it prints the even-numbered columns first, shifts the paper to the right, and prints the odd-numbered columns.

The 649-300 printer uses continuous forms ranging from four inches to 20.5 inches in width, and up to 22 inches (per page) in length.

640-102 Integrated Printer

The 640-102 integrated printer is housed in two cabinets; one cabinet contains the printer mechanisms, the other contains the printer logic circuitry (all electrical power is supplied by the processor). The printer's typeline is a revolving steel drum with alphanumeric characters embossed on its surface; there are 132 columns across the length of the typeline with 64 characters in each column. The printer may be equipped with either a single-numeric 64-character typeline or a double-numeric 64-character typeline. The single-numeric typeline contains 64 unique characters, including the letters A through Z, the numerals 0 through 9, and 28 "special" characters (mathematical, currency, and punctuation symbols). The double-numeric typeline contains 51 unique characters: The alphabetic character set, two numeric character sets, and 15 "special" characters (the comma, decimal point, and minus characters are included in the numeric character set and are repeated along with the numerals 0 through 9).

The typeline is divided into even- and odd-numbered columns, with all characters for the even-numbered columns on one half of the drum's circumference and the characters for the odd-numbered columns on the other side. Each print hammer covers two print positions; thus, to print a 132-character line of alphanumeric characters, each print hammer must be fired twice: once to print the even-numbered column and once to print the odd-numbered column. In the double-numeric mode (printing numerics only with the double-numeric typeline), each print hammer fires four times on each revolution of the typeline, thereby enabling the system to print two lines per revolution. Since the printline revolves at 450 revolutions per minute, the printer can print a maximum of 450 lines per minute using the single-numeric typeline, 900 lines per minute using the double-numeric typeline (printing numerics only). Actual printing speeds vary, depending on the number of lines skipped and the type of data printed (numerics, alphabetics, or both).

I/O WRITER

The basic NCR Century 101 System includes, as an integrated peripheral at position 2 of trunk 0, the thermal-printing I/O Writer. However, the system may contain the Teletype I/O Writer as an option. The I/O Writer enables the processor to display printed software messages or user messages to the operator and permits him to respond via keyboard entry.

The I/O Writer (thermal or Teletype) is selected by a 2-character Peripheral Address Field (PAF). The first character of the PAF selects the I/O Writer at trunk 0, position 2; the second character determines the function to be performed (reset, input permit, or output). The reset function sets the input permit flag OFF and puts the peripheral in an idle mode. The input permit function sets the input permit flag ON and puts the peripheral into an input mode, enabling the operator to input information to the processor via the keyboard. The output function places the peripheral into an output mode, enabling the processor to display a message to the operator. After selection, the I/O Writer uses peripheral control word 3 at memory location 0418 (hex) to (1) store the peripheral status character, (2) locate the next memory address to be accessed, and (3) determine when to terminate the I/O operation.

The following explanations describe the basic features of both I/O Writers.

Thermal I/O Writer

The thermal I/O Writer contains an ASCII keyboard and a friction platen on which forms can be printed. Printing is accomplished by a moveable print head that contains a 5 x 7 matrix of individually energized heating elements. When lightly pressed against heat-sensitive paper, the energized elements in the print head produce a dot pattern that resembles a printed character. This unit is capable of printing up to 94 different character patterns across an 80-column print line.

The keyboard for the thermal I/O Writer is a solid-state device containing 55 keys, which are capable of coding 128 characters, including numeric characters (0-9), upper- and lower-case alphabetical characters (A-Z), and special characters (mathematical, currency, and punctuation symbols). The keyboard also enables the user to perform all of the necessary control functions such as carriage return, line feed, and backspace.

The transfer rate of the thermal I/O Writer is 30 characters per second.

Teletype I/O Writer

The teletype I/O Writer contains an ASCII keyboard and a pin-feed platen on which 3-part forms can be printed; the method of printing and the keyboard vary greatly from the thermal I/O Writer, however.

Printing is accomplished by a Teletype-style moveable print cylinder that revolves to the desired character and then makes the impression by carrying the paper and ink ribbon against the platen.

The keyboard contains 60 keys, including a set of numeric characters, a set of alphabetic characters, a set of special characters, a line feed key, a carriage return key, a delete key, and an end-of-line bell key.

The transfer rate of the Teletype I/O Writer is six characters per second.

CRT I/O CONSOLE

The NCR Century 101 System may include, in addition to the I/O Writer, the optional 796-101 CRT (Cathode-Ray Tube) display console for communication between the operator and the processor. A console switch directs such communication to the I/O Writer or to the CRT. (The I/O Writer and the CRT cannot be used at the same time; there is no hard-copy backup.)

The 796-101 CRT display console has a 12-inch (diagonal) screen, which can accommodate up to 24 lines of data, 80 characters per line. Each character is formed by a 5 x 7 dot matrix. The character set of the CRT contains 64 ASCII characters, including alphabetic characters (A-Z), numeric characters (0-9), and special symbols.

The keyboard of the 796-101 is actually three separate keyboards: (1) a 54-key Teletype-compatible keyboard for alphanumeric input, (2) an 11-key numeric pad (0-9 and a decimal point) for rapid input of numeric data, and (3) a control array for positioning of the cursor on the screen, inserting or deleting characters, and controlling data entry and formatting.

The 796-101 CRT has several transfer rates, selectable by a switch; the data transfer rate may be 110, 300, 1200, 2400, or 9600 baud (asynchronous). The CRT refreshes the data on the screen at the rate of 60 frames per second.

TIME OF DAY CLOCK

The optional time of day clock (TOD) is based on a crystal oscillator and decoding matrix, which adds a binary one (called a "tick") to a 24-bit hardware register at memory locations 265, 266, and 267 every 10 milliseconds. When the clock register reaches 8 639 999 ticks (24 hours), it is reset to zero. The TOD clock is accurate to within $\pm$ 0.01%.

To obtain the time of day, the contents of the memory word at locations 264 through 267 (byte 264 always contains zero) must be read, then converted to hours, minutes, and seconds by software.

NOTE

Although all three bytes of the clock register are incremented simultaneously (by carry), a separate operation is normally required to read each byte. It is therefore advisable to move the contents of the clock register to another location, then compare that number of ticks against the current contents of the register to make sure that the clock register has not been incremented between read operations, which can cause an error of up to eleven minutes, depending on the location of the carry. If the two values

are not equal, the read and compare operations should be repeated until the transferred clock value is equal to the current clock value, before translating the obtained number of ticks into hours, minutes, and seconds.

Any attempt to write to memory locations 264 through 267, either by command or by console entry, causes the TOD clock to be reset to zero.

HARDWARE MULTIPLY/DIVIDE COMMANDS

The NCR Century 101 Processor may be equipped with the hardware MULTIPLY and DIVIDE commands to provide greater processing efficiency than conventional software methods which use repeated addition and subtraction to perform the multiply and divide functions. Both commands must be installed at the same time because the DIVIDE command uses some of the MULTIPLY logic.

The following explanations are functional descriptions only. For coding and programming specifications, see the NCR Century 101 Hardware Commands and Command Timing publication in this manual.

Hardware MULTIPLY Command

The MULTIPLY command multiplies the contents of the B field (multiplicand) by the contents of the A field (multiplier) and stores the result (product) in the memory accumulator. The command assumes the contents of the A and B fields to be signed, packed, decimal data with the signs stored on the rightmost four bits of each field. The maximum length of each field is eight bytes as designated by the T character of the command; TA (the rightmost four bits of the T character) indicates the length of the A field, and TB (the leftmost four bits of the T character) indicates the length of the B field. The final product is right-justified and stored in the memory accumulator (memory location 0140 - 014F) with the appropriate sign in the rightmost four bits of location 014F. A positive sign (1011) is stored if the signs of the A and B fields are equal; otherwise, a negative sign (1101) is stored.

The hardware multiplication process is similar to the manual arithmetic process; for example, each byte of the multiplicand (B field) is multiplied by each digit of the multiplier (A field), and the result (partial product) is shifted to the left one position before being added to the previous partial product. Sign characters are ignored during the multiplication process.

156	+	B-Field Contents (2 Bytes)
X 242	+	A-Field Contents (2 Bytes)
312		Partial product of first multiplication
624		Partial product of second multiplication
312		Partial product of third multiplication
37752	+	Final product

The processor's arithmetic logic unit performs the multiplication function one byte at a time, using a pair of multiplication tables in the processor's read-only memory. The multiply function is performed in cycles; during each cycle, the entire B-field is multiplied by one A-field digit and the result is stored in the memory accumulator (hex location 0140 - 014F). For example, during the first multiply cycle, the first B-field byte (two BCD values) is multiplied by the first A-field digit (one BCD value), and the result is stored in the rightmost byte of the memory accumulator. The next B-field byte is then multiplied by the same A-field digit, and the result is stored in the next available byte of the memory accumulator, thereby building the partial product in the accumulator one byte at a time. The first multiply cycle is complete when the last B-field byte has been multiplied by the A-field digit and the resulting value has been placed into the memory accumulator. The second multiply cycle begins with the next A-field digit and is completed when the last B-field byte has been multiplied by that digit; the result of each multiplication is added to the partial product already in the memory accumulator, beginning with the rightmost byte of the accumulator plus one. When the B-field has been multiplied by the last A-field digit, the command terminates, leaving the final product (right-justified) in the memory accumulator.

Hardware DIVIDE Command

The DIVIDE command divides the contents of the B-field (dividend) by the contents of the A-field (divisor) and stores the result (quotient) in the memory accumulator. The command assumes the contents of the A and B fields to be signed, packed, decimal data with the signs stored in the rightmost four bits of each field. The maximum length of the divisor (A field) is 8 bytes, and the maximum length of the dividend (B field) is 16 bytes. Divisor and dividend field lengths are both designated by the T character of the command; TA (the rightmost four bits of the T character) indicates the length of the divisor, and TB (the leftmost four bits of the T character) indicates the length of the dividend. The quotient and remainder are stored in the memory accumulator, which is evenly divided so that a maximum area of 8 bytes (15 digit positions plus a sign) is available for the quotient and a like area is available for the remainder.

The number of bytes needed for the quotient field in a given command is determined by subtracting TA (the length of the divisor) from TB (the length of the dividend); if the result of this subtraction is greater than eight or negative a program error occurs. Since the quotient field may be less than eight bytes in length, it is right-justified in the rightmost eight bytes of the memory accumulator. The quotient value is placed within the quotient field, right-justified and zero-filled to the left; the sign character occupies the rightmost four bits of the field.

The number of bytes needed for the remainder field in a given command is equal to the value of TA (length of the divisor); since this field may also be less than eight bytes in length, it is left-justified in the leftmost eight bytes of of the memory accumulator. The remainder value is left-justified within the remainder field and zero-filled to the right; however, if left-justifying the remainder results in the sign character being placed in the rightmost four bits of a byte, the remainder is shifted right one digit position and a leading zero is added.

Before division occurs, the command logic determines whether the quotient will fit within the allotted number of bytes in the memory accumulator. A program error is indicated if it is determined that the quotient will not fit within the allotted area.

Division is achieved by a trial quotient method similar to that used in manual division. For example, the logic circuitry compares the leftmost non-zero digit of the divisor to the leftmost non-zero digit of the dividend.

```
04230 | 00006791
```

If the divisor digit is equal to or less than the dividend digit, the logic circuitry assigns the appropriate trial quotient (4 goes into 6 once). The entire divisor value is then multiplied by the quotient value, and the resulting product is subtracted from the dividend.

```
          1   True quotient
04230 | 00006791
        4230
        2561  Remainder
```

If the divisor digit is larger than the dividend digit (04230 | 00326047), the logic circuitry assumes a dividend digit equal to 10 times its original value, plus nine (30 + 9 = 39). The trial quotient is then computed on the basis of the assumed dividend value (4 goes into 39 nine times). The entire divisor value is then multiplied by the trial quotient value, and the resulting product is subtracted from the dividend. If the subtraction results in a negative value, the trial quotient value is too large; if it results in a positive value, the trial quotient value becomes the true quotient value.

```
          9
04230 | 00326047
          38070
        9994534
```

When the trial quotient value is too large, the processor decrements the trial quotient by one and adds the entire divisor value to the negative result. This process is repeated until the remainder becomes positive again; at that time, the trial quotient value becomes the true quotient value.

```
          7   True quotient value
          8   Trial quotient value
          9   Trial quotient value
04230 | 00326047
          38070
        9994534  Negative value (trial quotient of 9)
          4230
        9998764  Negative value (trial quotient of 8)
          4230
        0002994  Positive value (trial quotient of 7)
```

After the first true value has been determined, the processor repeats the process, using the first non-zero digit of the divisor and the first non-zero digit of the positive value, plus the next digit of the dividend.

```
                 7
                 8
                 9
04230 | 00326047
          38070
        9994534
           4230
        9998764
           4230
        00029947
```

The DIVIDE command uses the read-only-memory (ROM) multiplication tables when multiplying the divisor by the trial quotient digit. The trial quotient digit, however, is obtained from a ROM divide table; the leftmost non-zero digit of the divisor and the leftmost non-zero digit of the dividend are input to this table to arrive at the trial quotient digit. The divide table (shown in the following illustration) considers two situations: (1) the divisor digit is greater than the dividend digit, and (2) the divisor digit is equal to or less than the dividend digit. For example, if the divisor digit is 4 and the dividend digit is 6, the trial quotient digit is 1, but if the divisor digit is 6 and the dividend digit is 4, the trial quotient is 8 (the table assumes the dividend digit to be 49 instead of 4).

		LEFTMOST NON-ZERO DIVISOR DIGIT								
		1	2	3	4	5	6	7	8	9
LEFTMOST NON-ZERO DIVIDEND DIGIT	1	1	9	6	4	3	3	2	2	2
	2	2	1	9	7	5	4	4	3	3
	3	3	1	1	9	7	6	5	4	4
	4	4	2	1	1	9	8	7	6	5
	5	5	2	1	1	1	9	8	7	6
	6	6	3	2	1	1	1	9	8	7
	7	7	3	2	1	1	1	1	9	8
	8	8	4	2	2	1	1	1	1	9
	9	9	4	3	2	1	1	1	1	1

HARDWARE LOGIC COMMAND

The NCR Century 101 Processor may be equipped with the hardware LOGIC command, which enables the programmer to use the processor to perform logic operations on the contents of two 1-byte fields (A and B). The rightmost four bits of the command's T character (b4-b1) specify which of the 16 available functions is to be performed (the leftmost four bits of the T character are not used). The logic operation, based on Boolean algebra, uses the individual bit values of corresponding A and B field bit positions as variables for the equations; the result of the operation is stored in the B field.

The following table indicates the logic functions that may be performed and specifies the T value necessary to select each function. For an explanation of each function, see the LOGIC command description in this manual under the PROCESSORS tab, "NCR Century 101 Hardware Commands and Command Timing."

FUNCTIONS OF LOGIC COMMANDS

T (b_4-b_1) HEX VALUE	FUNCTION
0	$\overline{B}$ = 0 (clear B)
1	$\overline{B}$ = (A + B)' or A'B'
2	$\overline{B}$ = A'B
3	$\overline{B}$ = A'
4	$\overline{B}$ = AB'
5	$\overline{B}$ = B'
6	$\overline{B}$ = AB' + A'B
7	$\overline{B}$ = (AB)' or A' + B'
8	$\overline{B}$ = AB
9	$\overline{B}$ = A'B' + AB
A	$\overline{B}$ = B (store B)
B	$\overline{B}$ = A' + B
C	$\overline{B}$ = A (store A)
D	$\overline{B}$ = A + B'
E	$\overline{B}$ = A + B
F	$\overline{B}$ = 1 (set B)

ONLINE COMMUNICATIONS

The NCR Century 101 system can be equipped for online communication by adding the integrated communications interface (feature no. 6002) to position 3 of trunk 0, or by adding the 621-103 Communications Multiplexor to any available low-speed common-trunk position. The user may begin online processing with the integrated communications interface, which permits a limited number of adapters. Later, he may graduate to the free-standing multiplexor as his system requirements increase. The interface and the multiplexor are functionally similar. While the system does not normally include both devices, the interface could serve as a back-up peripheral for the multiplexor. A

complete description of the 621-103 Communications Multiplexor and the integrated communications interface feature can be found in this manual under the ONLINE COMMUNICATIONS tab, "621-103 Communications Multiplexor". Subsequent publications under the same tab describe the adapters that may be used.

The integrated communications interface and a maximum of seven or ten adapters can be installed in the processor's cabinet. While there is space for ten adapters (five boards), the fact that a total of sixteen control words may be associated with the interface influences the number of adapters permitted. When all adapters require one control word, ten adapters and the interval timer are permitted. (The interval timer is an integral part of the interface.) In this case, a total of eleven control words are used. However, when any adapter requires two control words, the interface effectively requires two control words for each adapter and the timer. Thus, of the sixteen control words available for the interface, fourteen control words can be assigned to seven adapters and two control words must be reserved for the timer. For a detailed description of adapter addressing and control word use, see the ONLINE/REAL TIME PROGRAMMING MANUAL, APPENDIX tab, "Control Word Assignment and PAL Construction for Online Systems". That publication discusses requirements of both the integrated communications interface and the 621-103 Communications Multiplexor.

The basic model of the 621-103 multiplexor permits installation of fifteen adapters and the interval timer. Logic extensions added to the basic model permit a maximum of 253 adapters which may be installed in the multiplexor cabinet and in auxiliary bays. (This maximum figure is subject to practical considerations which may reduce the actual number of adapters permitted.) To efficiently service a large number of adapters, the 621-103 multiplexor provides the hardware-assisted software queueing (HASQ) feature. The HASQ feature reduces control word scanning and thus speeds service to adapters. This feature is explained in detail in this manual under the ONLINE COMMUNICATION tab, "621-103 Communications Multiplexor".

MULTIPROGRAMMING FEATURE

Multiprogramming is a technique whereby two or more programs, concurrently resident in memory, can share the time and resources of a computer.

Generally, a program is forced to wait for the completion of each I/O operation, since the external devices are much slower than the central processor. It is therefore advantageous to have more than one program resident in memory, managed in such a way that while some programs wait for the completion of their I/O requests, others can use the resources of the central processor.

By reducing the processor's idle time, multiprogramming significantly increases the productivity of the system. Because the processor's resources are not devoted to a single program, the NCR Century 101 system can provide economical realtime, online services to many users.

Multiprogramming is an optional hardware feature of the NCR Century 101. The following are prerequisites for the implementation of this feature:

- Extended Addressing Logic (EAL) feature
- At least 64K bytes of memory
- LOGIC command
- I/O Writer
- Three discs (minimum)

To switch efficiently from one program to another, to allocate memory space, and to manage the operations of the various concurrent programs, the NCR Century 101 uses a combination hardware/software multiprogramming method. The discussion of multiprogramming that follows deals primarily with hardware features; software is mentioned only where required for clarity.

Program Switching

When switching from one program to another, the processor must save the status of the interrupted program, its index registers, and its private data. Switching programs in this manner causes a large software overhead.

The NCR Century 101 accomplishes program switching by use of both software and BAR/LAR hardware registers. A BAR (Base Address Register) contains the beginning (lowest) address of a program segment in memory, and a LAR (Limit Address Register) contains the length of that segment, thereby establishing the upper limit of the segment. The BAR and LAR are 7-bit registers, containing addresses in 2048-character units (that is, the BAR/LAR contents are treated as though they were followed by 11 zero bits).

When giving control to a new program, instead of saving and restoring the current program's index registers and data, the Executive program changes the contents of the BAR/LAR registers to refer to the memory area where the new program is located. This results in large savings in software overhead when switching programs.

Memory Allocation

The multiprogramming operating software determines memory space requirements, peripheral requirements, and other pertinent requirements of the program, according to the parameters entered during system initialization. Software then allocates the memory and the peripherals as efficiently as possible.

Since memory space is allocated dynamically and the addresses in a compiled program are relative to the contents of the BAR/LAR, the programmer may write his programs with the starting address relative to zero.

Supervisor/User States

For increased efficiency in executing multiple programs, the NCR Century 101 uses two modes of operation -- user and supervisor.

The processor is always in either the user state or the supervisor state, as determined by the supervisor flag (S-flag). If the S-flag is ON, the processor is in the supervisor state; if the S-flag is OFF, the processor is in the user state.

In the user state, the processor executes the customer programs normally. In the supervisor state, the processor handles special conditions that cannot be completed in the user state. Whenever the processor encounters conditions that require special handling, it turns ON the S-flag and enters the supervisor state. The following flows (and the initial powering-up sequence) cause the S-flag to be turned ON:

- Program Interrupt Trapping
- ME Trapping
- PE Trapping
- ICC Trapping
- Console Loading

When the processor resumes normal processing, a RESTORE command turns off the S-flag and the processor enters the user state.

Status Word

All trapping flows store the state of the S-flag in bit 8 of the ninth character of the Status Word. The S-flag is stored in the state it was in prior to entering the flow, ignoring any change that may occur during the flow. The contents of the ninth character of the Status Word are:

- bit 8 - S-flag
- bit 7 - Always OFF
- bit 6 - Overflow Flag
- bit 5 - Repeat Indicator
- bit 4 - Always OFF
- bit 3 - Greater Flag
- bit 2 - Equal Flag
- bit 1 - Less Flag

The remaining characters of the Status Word are shown in the "NCR Century 101 Reserved Memory Areas" illustration earlier in this publication.

Privileged Commands

Certain commands, called Privileged Commands, may be executed only in the supervisor state, because they require special handling under software control. An attempt to execute a privileged command in the user state causes the processor to take the ICC Trap. The following are privileged commands:

- INOUT
- WAIT
- SWITCHES INPUT
- LOAD BAR

The IP ON and IP OFF commands, although not privileged commands, affect the state of the hardware IP indicator in the supervisor state only. In the user state, these commands operate on a pseudo IP indicator, located in bit 1 of absolute memory location 272, instead of the actual IP indicator.

BAR/LAR usage

- Relative Addressing

  Each program contains the A and B addresses and the addresses of the control registers used by the program in "relative" form, rather than in absolute form. If required, addition of the BAR contents to these addresses occurs just prior to a memory access. The A, B, or CR values stored in the Error and Program Status Words are unaffected by the BAR contents. (The Error Status Words, Program Status Words, and Error Code Characters are stored in a system software area; their addresses are not subject to BAR/LAR manipulation.)

- LAR Check

  When a memory access is to be performed under BAR/LAR processing, the contents of the LAR are compared to bits 12-18 of the virtual effective address. If the LAR contents are equal to zero or are greater than the contents of bits 12-18, the BAR value is added to the virtual effective address and the memory access is permitted. If the virtual effective address is equal to or greater than the LAR contents, a PE occurs.

  Before executing a command to transfer control, the processor compares the A address to the current contents of the LAR. If the A address is equal to or greater than the LAR value, and the conditions for transfer of control are satisfied, a PE occurs; the control register for the program remains unaltered. If the A address is less than the LAR value, or if LAR is equal to zero, the command execution proceeds normally.

  When a RESTORE command is to return control to the user state, the address that is to be placed in the control register is compared to the current contents of the LAR. If the address is greater than or equal to the LAR value, a PE occurs; the control register remains unaltered. If the address to be placed in the control register is less than the LAR value, execution of the RESTORE command proceeds normally.

- BAR Addition

  In memory accesses pertaining to command setup and execution under control of BAR/LAR processing, when the LAR check indicates that the memory access can proceed, the contents of the BAR are added to bits 12-18 of the virtual effective address to obtain the absolute address to be accessed. A carry out of the 18th bit position causes an immediate PE.

Effects of BAR/LAR in the User State

In the user state (S-flag OFF), all command setup and execution is subject to BAR/LAR processing.

Effects of BAR/LAR in the Supervisor State

In the supervisor state (S-flag ON), BAR/LAR processing is dependent on two hardware flags: Flag A for the A value and Flag B for the B value.

If, during the setup phase, the RA character of the command refers to an index register from 1 through 31, Flag A is set ON; otherwise Flag A is set OFF. If Flag A is ON, the index register referred to in the RA portion of the command is a user index register relative to the current BAR contents. RA characters obtained by mode 1 indirect addressing have no effect on Flag A.

The control of Flag B is identical to that of Flag A, except that values from the B portion of the command are used. (Note that commands which have no B portion have no effect on Flag B. Thus, if Flag B is ON, it will remain in that state until acted upon by the next double-stage command. The programmer should take this into consideration when chaining commands, in the event that a control transfer to a subroutine or a trapping routine occurs between double-stage commands.)

All memory references by the A or B values, including those derived by indirect addressing, are subject to BAR/LAR processing if the associated Flag A or Flag B is ON. If the associated flag is OFF, BAR/LAR processing does not take place for that value.

The following special locations are addressed as absolute index registers, regardless of the state of the S-flag, Flag A, and Flag B.

- The Error Status Word in IR 5-7
- The Program Status Word in IR 10-12
- The Error Code Character in IR 9
- The pseudo II and IP in the Interrupt Control Word

The following trapping flows transfer control to an absolute address, regardless of the state of the S-flag.

- Program Interrupt Trapping Flow
- ME Trapping Flow
- PE Trapping Flow
- ICC Trapping Flow

The S-flag has no effect on the I/O control flows.

LOAD BAR Command

The processor must be in the supervisor state to execute the LOAD BAR command. The A operand address of the LOAD BAR command specifies a 4-character field, of which only the rightmost two characters are used. The low-order seven bits of the first (left) of these two characters contain the value to be placed in the BAR; the low-order seven bits of the second character contain the value to be placed in the LAR. Bit 8 is not used in either character.

If the A address is not evenly divisible by four (0 mod 4), a PE occurs; the BAR/LAR registers remain undisturbed.

The T and B portions of the LOAD BAR command are not used.

Interval Timer

Included in the Multiprogramming Feature is an interval timer, which provides the operating system with the ability to interrupt a program after a specified number of milliseconds. Thus, in a multiprogramming environment, the interval timer prevents any program from using more processor time than specified. By doing so, the interval timer also detects and prevents program loops.

The interval timer makes use of certain input/output features in its operation. Once every millisecond, the timer requests service from the I/O control. The I/O control reads the NA portion of the timer's control word, increments it by one, and compares the incremented NA to the TA portion of the control word. If NA is not equal to TA, the I/O control writes the incremented NA back into the control word and the timer continues to count and raise service requests to the I/O control. When NA equals TA, an Operation Complete S3 status character is stored in the interval timer's control word and the Interrupt Indicator is turned ON unconditionally (no priority check is performed). The interval timer uses a special control word at memory location 336. The format of this control word is identical to any other control word.

The interval timer continues to count and raise service requests until one of the following conditions occurs: NA equals TA, the I/O control fails to service a request, the processor enters the Halt state, or a latent ME is detected.

When NA equals TA, an Operation Complete S3 status character (0000 0000) is stored in the control word and the Interrupt Indicator is turned ON unconditionally.

Each 1-millisecond count (called a "tic") of the interval timer raises a service request to the I/O control for 750 microseconds. If the request is not serviced within this time, the subsequent request causes a System Overload S3 status character (0001 0000) to be stored in the control word and the Interrupt Indicator to be turned ON unconditionally.

The interval timer runs continuously, except when the processor is in the Halt state. When the processor enters the Halt state, a Special S3 status character (0000 0001) is stored in the control word, the Interrupt Indicator is turned ON unconditionally, and the timer stops counting. The timer does not resume counting until the processor is taken out of the Halt state, with the first "tic" occurring not less than one, nor more than two, milliseconds after the processor leaves the Halt state.

If a latent ME is detected, an S4 status character (1000 0100) is stored in the control word and the Interrupt Indicator is turned ON unconditionally.

The basic 1-millisecond increment of the interval timer is accurate to within 0.01%. The timer's NA can be counted as high as 65,535; it is then restored to zero. Therefore, the maximum time between interrupts caused by NA being equal to TA is 65.536 seconds.

NCR CENTURY 101 HARDWARE COMMANDS

AND COMMAND TIMING

This publication, intended as a supplement to the "NCR Century 101 Processor" publication, contains a functional description of all the hardware commands available for the NCR Century 101. There are 41 hardware commands available, including seven that are optional. The following table lists the commands, the mnemonics for the commands, and the command code representation; the command timing is located at the end of this publication.

HARDWARE COMMANDS

Command	Mnemonic	Code			Page
		Decimal Representation	Hexadecimal Representation		
			8 Byte Length	4 Byte Length	
ADD BINARY	BADD	96	60	E0	6
ADD SIGNED	PADD	64	40	C0	9
ADD UNSIGNED	UADD	98	62	E2	14
BRANCH EQUAL	BRE	106	6A	EA	17
BRANCH GREATER	BRG	108	6C	EC	18
BRANCH GREATER OR EQUAL	BRGE	110	6E	EE	19
BRANCH LESS	BRL	105	69	E9	20
BRANCH LESS OR EQUAL	BRLE	107	6B	EB	21
BRANCH LESS OR GREATER	BRU	109	6D	ED	22
BRANCH OVERFLOW	BROV	104	68	E8	23
BRANCH UNCONDITIONALLY	BR	111	6F	EF	24
COMPARE BINARY	BCOMP	101	65	E5	25
COMPARE SIGNED	PCOMP	69	45	C5	28
**COUNT	COUNT	74	4A	CA	32
DECODE ALL	DCODA	79	4F	CF	33
DECODE TO DELIMITER	DCODD	78	4E	CE	34
*DIVIDE	PDIV	113	71	F1	37
INOUT	INOUT	112	70	F0	41
JUMP	JUMP	75	4B	CB	43
*LOGIC	LOGIC	94	5E	DE	44
MOVE A LEFT TO RIGHT	MVAL	84	54	D4	49
MOVE A RIGHT TO LEFT	MVAR	100	64	E4	51
MOVE B RIGHT TO LEFT	MVBR	68	44	C4	54
*MULTIPLY	PMULT	93	5D	DD	56
OPTION SWITCHES INPUT	SWIN	80	50	D0	59
PACK	PACK	76	4C	CC	59
REPEAT	REPEAT	102	66	E6	63
RESTORE	RESTOR	72	48	C8	65
**SCAN ON KEY EQUAL	SCANE	86	56	D6	67
**SCAN ON KEY GREATER THAN	SCANG	87	57	D7	68
**SCAN ON KEY LESS THAN	SCANL	85	55	D5	69
SET IP OFF	IPOFF	71	47	C7	70
SET IP ON	IPON	70	46	C6	71
SUBTRACT BINARY	BSUB	97	61	E1	72
SUBTRACT SIGNED	PSUB	65	41	C1	74
SUBTRACT UNSIGNED	USUB	99	63	E3	79
TEST BIT	TESTB	83	53	D3	83
TEST CHARACTER EQUAL	TESTCE	81	51	D1	85
TEST CHARACTER UNEQUAL	TESTCU	82	52	D2	86
UNPACK	UNPACK	77	4D	CD	88
WAIT	WAITI	103	67	E7	91

* Indicates optional command
** Indicates optional command available only with Multiprogramming feature

COMMAND FORMAT AND OPERATION

A command may be either four or eight bytes in length, with the length being specified by the command code. The length of a command, however, is pertinent only to command setup; it does not effect command execution.

8-BYTE COMMAND FORMAT

The 8-byte command format (Q RA A2A1 T RB B2B1) is composed of four basic parts:

1. The command code (Q), which is one byte in length, designates the command to be performed and its length.
2. The A operand (RA A2A1), which is three bytes in length, designates the mode of addressing if any and the address of the A field.
3. The T character (T), which is one byte in length, is used for varying functions, depending upon the type of command.
4. The B operand (RB B2B1), which is three bytes in length, designates the mode of addressing if any and the address of the B field.

Command Code

By its bit configuration, the command code designates both the command to be executed and its length. The decimal value of bits 7 through 1 (as shown in the Hardware Commands Table) determines the command that is to be executed. Bit 8 (leftmost bit) of the command code, when OFF, designates the command to be eight bytes. For simplicity, the hexadecimal value of all eight bits is used to indicate the type of command (including length), and it is this value which appears on program listings and memory dumps.

A Operand

The 3-byte A operand, which is used to determine the effective A address (leftmost byte) of the A field, is composed of two parts: The associated index register (RA) and the partial A address (A2A1).

Bits 8 through 3 of RA indicate the binary address of the associated A index register; bits 2 and 1 indicate the mode of addressing.

MODES OF ADDRESSING		
MODE	RA	
	Bit 2	Bit 1
Direct Addressing	0	0
Mode 1 Indirect Addressing	0	1
Mode 2 Indirect Addressing	1	0
Mode 3 Incremental Addressing	1	1

For simplicity, the hexadecimal value of all eight bits of RA indicate both the associated index register and the mode of addressing; it is this value which appears on program listings and memory dumps. (Refer to PRODUCT

INFORMATION REFERENCE MANUAL, PROCESSORS, "NCR Century 101 Processor," under the heading of Addressing Memory By Hardware Command for further information pertaining to modes of addressing.)

The A2A1 portion of the A operand (two bytes) contains the partial A address, which is added to the contents of the associated index register (if any) to arrive at either the effective A address or the indirect A address, depending on the mode of addressing. The indirect A address points to another area in memory that may contain either another indirect A address or the effective A address.

T Character

The use of the T character (one byte) varies according to the command; however it normally indicates the lengths of both the A and B fields. All functions of the T character are explained in the command descriptions that follow.

B Operand

The B operand is identical to the A operand, except that the B operand is used to determine the effective address of the B field.

8-BYTE COMMAND OPERATION

Functionally, the processor hardware uses two distinct phases in performing an 8-byte command: (1) the setup phase, and (2) the execution phase.

The command setup phase, which is common to all commands, must be completed before command execution and before the trapping flows can be performed. During command setup, the command is read from memory, the modes of addressing and the value of the T character are determined, and the effective A and effective B addresses are computed. The processor also checks at this time for various illegal functions, such as illegal command codes and effective addresses greater than memory size. On completion of command setup, (1) the effective A address, effective B address and T character are in the appropriate hardware registers where they are available for command execution, (2) indexing and indirect addressing have been completed and, if mode 3 addressing is used, the effective addresses are stored in the associated index registers, and (3) the control register is incremented by eight to point to the next command in sequence.

During the command execution phase, the processor performs the function specified by the command code. On completion of the execution phase, the implied T character and implied B address are stored in hardware registers where they are available to subsequent 4-byte commands. The implied T character always has the same value as the T character prior to command execution. The implied B address, however, varies according to the command that has been executed (the method used to establish the implied B address is included in the following command descriptions).

4-BYTE COMMAND FORMAT

The 4-byte command format (Q RA A2A1) is composed of two basic parts:

1. The command code (Q), which is one byte, designates the command to be performed and its length.
2. The A operand (RA A2A1), which is three bytes in length, designates the mode of addressing if any and the address of the A field.

Command Code

The command code designates (by its bit configuration) both the command to be executed and its length. The decimal value of bits 7 through 1 (as shown in the Hardware Commands Table) determines the command that is to be executed. Bit 8 (leftmost bit) of the command code, when ON, designates the command to be four bytes. For simplicity, the hexadecimal value of all eight bits is used to indicate the type of command (including length), and it is this value which appears on program listings and memory dumps.

A Operand

The A operand for the 4-byte command is identical in structure and operation to the 8-byte command A operand.

4-BYTE COMMAND OPERATION

To perform a 4-byte command, the processor hardware uses a setup phase, the implied T and B values, and an execution phase.

Command Setup

During command setup, the processor reads the command from memory, determines the mode of addressing for the A operand, and computes the effective A address. The processor also checks for various illegal functions at this time.

Upon completion of the command setup, (1) the effective A address is available for command execution, (2) indexing or indirect addressing is completed for the A operand and, if mode 3 addressing is used, the effective A address is stored in its associated index register, (3) the control register is incremented by 4 and points to the next command in sequence. The T character and effective B address, which are available from the previously executed 8-byte command as implied values, are not setup by this command.

Implied T Character

All commands, whether four or eight bytes, terminate execution with an implied T character in a hardware register, where it is available to subsequent 4-byte commands. The implied T character always has the same value as the T character prior to command execution; therefore, the T character established by the previous 8-byte command is used during the execution of all subsequent 4-byte commands as if those commands had set them up.

Implied B Address

All 8-byte commands terminate with an implied B address in a hardware register, where it is available to subsequent 4-byte commands. The 4-byte command uses the implied B address, during execution, as an effective B address and in the same manner as the 8-byte command uses an effective B address. Upon termination

of the 4-byte command, an implied B address is available to yet another 4-byte command.

The implied B address may vary after execution, depending upon the command being executed; however, it is normally the same as the effective B address prior to execution. The methods used to establish the implied B address are included in the following command descriptions.

Because the effective B address is not initially setup by the 4-byte command, only direct addressing is available.

Command Execution

The command execution phase of the 4-byte command is identical to that of the 8-byte command execution phase.

HARDWARE COMMAND DESCRIPTIONS

All command descriptions assume the following:

1. A program has been compiled and is resident in memory in a form recognizable to processor hardware.
2. The command setup phase has been completed. (Refer to PRODUCT INFORMATION REFERENCE MANUAL, PROCESSORS, "NCR Century 101 Processor", under the heading of ALU Functional Operation.)

The values shown in the command format are hexadecimal.

ADD BINARY (BADD)

Command code: 96 (60) (EO)

The ADD BINARY command binarily adds the contents of the field specified by the effective A address to the contents of the field specified by the effective B address; the result replaces the original B field contents. The A and B fields are considered to contain unsigned, positive binary data. The lengths of both the A and B fields are indicated by the T character, which may vary in value from 0 through 255, with 0 indicating a length of 256 bytes.

Addition is performed from right to left, one byte at a time, with a carry from one byte being added to the byte on its left. A carry beyond the leftmost byte does not effect the state of the overflow flag and is not included in the result of the addition.

The result obtained when adding the contents of overlapping A and B fields may differ from the result obtained when adding the contents of A and B fields that do not overlap (assuming the A and B values are the same in both instances). Addition of overlapping fields normally changes the contents of the A field.

The implied B address for a subsequent 4-byte command is the same as the effective B address established prior to the execution of the ADD BINARY command.

Example 1

Q	RA	A2	A1	T	RB	B2	B1
60	37	10	00	03	38	00	00

Partial B address = 00000 (dec.)
Index register 14, mode 0
Field Length = 3 bytes
Partial A address = 04096 (dec.)
Index register 13, mode 3
Command code for an 8-byte ADD BINARY command

Assume:

Index register 13 contains 0000 (00000 dec.)
Index register 14 contains 1A00 (06656 dec.)

After command setup:

Effective A address = 1000 + 0000 = 1000 (04096 dec.)
Effective B address = 0000 + 1A00 = 1A00 (06656 dec.)

Before command execution:

A field address (hex.)	1000	1001	1002
A field contents (bin.)	10101100	11001010	00101101

B field address (hex.)	1A00	1A01	1A02
B field contents (bin.)	11100110	10111110	01110101

Arithmetic manipulation:

A field contents	10101100	11001010	00101101
B field contents	11100110	10111110	01110101
	01001010	01110100	01011000
Carries	1 1 1	111111	11111 1
Final result	1 10010011	10001000	10100010

The carry beyond the specified field length is ignored.

Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	1A00	1A01	1A02
B field contents (bin.)	10010011	10001000	10100010

The implied B address for a 4-byte command following the execution of this ADD BINARY command is 1A00 (06656 dec.).

Example 2 (Fields overlap)

Q	RA	A2	A1
E0	36	00	30

Partial A address = 00048 (dec.)
Index register 15, mode 0
Command code for a 4-byte ADD BINARY command

Assume:

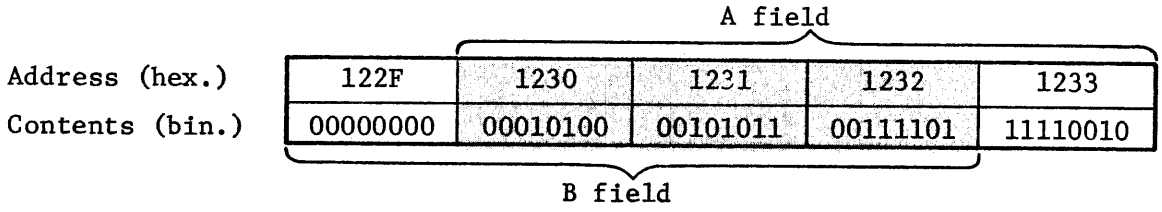
Index register 15 contains 1200 (04608 dec.)

After command setup:

Effective A address = 0030 + 1200 = 1230 (04656 dec.)

The effective B address and the T value were established by a previously executed 8-byte command. Assume that the T character has a value of 4, and the implied B address is 122F (04655 dec.).

Before command execution:



Arithmetic manipulation (overlapping fields):

1. The contents of address 1233 are added to the contents of address 1232, and the result is stored in 1232.

```
 11110010
 00111101
 11001111
 111          Carries
1 00101111    Final Result
```

Carry to next byte

2. The contents of address 1232 (altered by step 1) are added to the contents of address 1231, and the result is stored in 1231.

```
 00101111
 00101011
 00000100
  1 11111    Carries (including
 01011011    initial carry from
             step 1).
```

Final result

3. The contents of address 1231 (altered in step 2) are added to the contents of address 1230, and the result is stored in 1230.	01011011 00010100 01001111 1 01101111	 Carry Final Result
4. The contents of address 1230 (altered by step 3) are added to the contents of address 122F, and the result is stored in 122F.	01101111 00000000 01101111 01101111	 Carries (none) Final Result

Following command execution:

A field

Address (hex.)	122F	1230	1231	1232	1233
Contents (bin.)	01101111	01101111	01011011	00101111	11110010

B field

The implied B address for a 4-byte command following the execution of this ADD BINARY command is 122F (04655 dec.).

ADD SIGNED (PADD)

Command code: 64 (40) (C0)

The ADD SIGNED command decimally adds the contents of the field specified by the effective A address to the contents of the field specified by the effective B address; the result replaces the original B field contents. The initial contents of the A and B fields are considered to be signed, packed (see PACK command description) decimal information with the sign stored in the rightmost four bits of each field. Fields with negative values are indicated by the bit configuration 1101; any other configuration indicates the field to contain a positive value. The lengths of both the A and B fields are indicated by the T character, which may vary in value from 0 through 255, with 0 indicating a length of 256 bytes.

CAUTION

Do not add hexadecimal fields; the result is predictable, but not a correct hexadecimal sum.

Addition is performed from right to left, one byte at a time, with a carry from one byte being added to the byte on its left. A carry beyond the leftmost byte is not included in the result; however, this carry sets the overflow flag ON. If a carry is not generated beyond the leftmost byte, the initial state of the overflow flag is preserved. The result of the addition replaces the original B field contents.

The result obtained when adding the contents of overlapping A and B fields may differ from the result obtained when adding the contents of A and B fields that

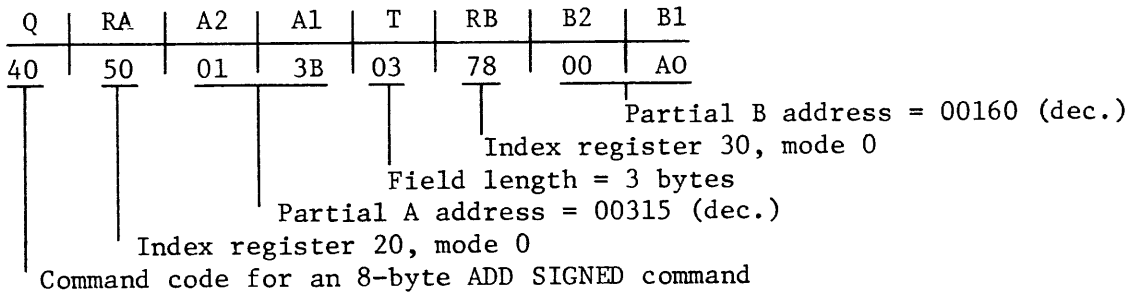
do not overlap (assuming the A and B values are the same in both instances). The addition of overlapping fields normally changes the contents of the A field.

The B field, following command execution, normally assumes the sign of the field with the greatest absolute value (algebraic sign derivation); if the absolute values of both the A and B fields are equal, the initial B field sign is retained. The appropriate bit configuration (1011, indicating a positive value or 1101; indicating a negative value) is stored in the B field if a sign change occurs.

The implied B address for a subsequent 4-byte command is the same as the effective B address established prior to the execution of the ADD SIGNED command.

The following examples show the three methods used by the ADD SIGNED command to determine the results of the addition.

Example 1 (Like signs)



Assume:

Index register 20 contains 0FFE (04094 dec.)
Index register 30 contains 10E0 (04320 dec.)

After command setup:

Effective A address = 013B + 0FFE = 1139 (04409 dec.)
Effective B address = 00A0 + 10E0 = 1180 (04480 dec.)

Before command execution:

A field address (hex.)	1139		113A		113B	
A field contents (BCD)	0000	0111	0110	0101	0100	1011
Decimal equivalent	0	7	6	5	4	+

B field address (hex.)	1180		1181		1182	
B field contents (BCD)	0000	0001	0000	0001	0001	1011
Decimal equivalent	0	1	0	1	1	+

Arithmetic manipulation:

Decimal equivalent of addition to be performed:

```
 07654
 01011
 08665
```

BCD addition:

1. Excess 6 is added to each A field digit after it is read from memory.

A field digits	0000	0111	0110	0101	0100
Excess 6	0110	0110	0110	0110	0110
	0110	0001	0000	0011	0010
Carries		11	11	1	1
Result	0110	1101	1100	1011	1010

2. The B field digits are added to the results obtained in step 1.

Results of step 1	0110	1101	1100	1011	1010
B field digits	0000	0001	0000	0001	0001
	0110	1100	1100	1010	1011
Carries		1		1	
Uncorrected result	0110	1110	1100	1100	1011

3. A binary ten is added to the uncorrected result of any digit addition (from step 1 or 2) which did not generate a carry to the digit on its left (including the leftmost digit).

Uncorrected result	0110	1110	1100	1100	1011
Binary ten correction	1010	1010	1010	1010	1010
	1100	0100	0110	0110	0001
Carries	1 11	1 11	1	1	1 1
Final result	0000	1000	0110	0110	0101

Carries between digits are ignored.

Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	1180		1181		1182	
B field contents (BCD)	0000	1000	0110	0110	0101	1011
Decimal equivalent	0	8	6	6	5	+

The implied B address for a 4-byte command following the execution of this ADD SIGNED command is 1180 (04480 dec.).

Example 2 (Unlike signs with A field absolute value greater than B field absolute value)

Assume that the fields used in the preceding example contain the following information before command execution:

A field address (hex.)	1139		113A		113B	
A field contents (BCD)	0000	0111	0110	0101	0100	1011
Decimal equivalent	0	7	6	5	4	+

B field address (hex.)	1180		1181		1182	
B field contents (BCD)	0000	0100	0001	0101	0110	1101
Decimal equivalent	0	4	1	5	6	-

Arithmetic manipulation:

Decimal equivalent of addition to be performed: 07654+
04156-
03498+

BCD addition:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost digit of the A fields 1's complement.

1's complement	1111	1000	1001	1010	1011
Initial carry					1
	1111	1000	1001	1010	1010
Carries					11
2's complement	1111	1000	1001	1010	1100

2. The B field digits are added to the 2's complement of the A field.

2's complement	1111	1000	1001	1010	1100
B field digits	0000	0100	0001	0101	0110
	1111	1100	1000	1111	1010
Carries			11	1111	1
Uncorrected results	1111	1100	1011	0000	0010

3. A binary ten is added to the uncorrected result of any digit addition that did not generate a carry to the digit on its left (including the leftmost digit) from either step 1 or 2.

Uncorrected result	1111	1100	1011	0000	0010
Binary ten correction	1010	1010	1010		
	0101	0110	0001	0000	0010
Carries	1 11	1	1 1		
Corrected result	1001	0110	0101	0000	0010

Carries between digits are ignored.

4. The 2's complement of the corrected result from step 3 is formed.

1's complement	0110	1001	1010	1111	1101
Initial carry					1
	0110	1001	1010	1111	1100
Carry					1
2's complement	0110	1001	1010	1111	1110

5. A binary ten is added to the 2's complement (from step 4) of any digit addition that did not generate a carry to the digit on its left (including the leftmost digit).

2's complement	0110	1001	1010	1111	1110
Binary ten correction	1010	1010	1010	1010	1010
	1100	0011	0000	0101	0100
Carries	1 11	1	1 1	1 11	1 11
Final result	0000	0011	0100	1001	1000

Carries between digits are ignored.

Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	1180		1181		1182	
B field contents (BCD)	0000	0011	0100	1001	1000	1011
Decimal equivalent	0	3	4	9	8	+

The implied B address for a 4-byte command following the execution of this ADD SIGNED command is 1180 (04480 dec.).

Example 3 (Unlike signs with B field absolute value greater than A field absolute value)

Assume that the fields used in example 1 contain the following information before command execution:

A field address (hex.)	1139		113A		113B	
A field contents (BCD)	0000	0100	0001	0101	0110	1101
Decimal equivalent	0	4	1	5	6	-

B field address (hex.)	1180		1181		1182	
B field contents (BCD)	0000	0111	0110	0101	0100	1011
Decimal equivalent	0	7	6	5	4	+

Arithmetic manipulation:

Decimal equivalent of addition to be performed:

04156-
07654+
03498+

BCD addition:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost digit of the A fields 1's complement.

1's complement	1111	1011	1110	1010	1001
Initial carry					1
	1111	1011	1110	1010	1000
Carries					1
2's complement	1111	1011	1110	1010	1010

2. The B field digits are added to the 2's complement of the A field.

2's complement	1111	1011	1110	1010	1010
B field digits	0000	0111	0110	0101	0100
	1111	1100	1000	1111	1110
Carries	1 1111	1111	11		
Uncorrected result	0000	0011	0100	1111	1110

3. A binary ten is added to the uncorrected result of any digit addition that did not generate a carry to the digit on its left (including the leftmost digit) from either step 1 or 2.

Uncorrected result	0000	0011	0100	1111	1110
Binary ten correction				1010	1010
	0000	0011	0100	0101	0100
Carries				1 11	1 11
Final result	0000	0011	0100	1001	1000

Carries between digits are ignored

Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	1180		1181		1182	
B field contents (BCD)	0000	0011	0100	1001	1000	1011
Decimal equivalent	0	3	4	9	8	+

The implied B address for a 4-byte command following the execution of this ADD SIGNED command is 1180 (04480 dec.).

ADD UNSIGNED (UADD)

Command code: 98 (62) (E2)

The ADD UNSIGNED command decimally adds the contents of the field specified by the effective A address to the contents of the field specified by the effective B address; the result replaces the original B field contents. The initial contents of the A and B fields are considered to be ASCII characters zero through nine. The lengths of both the A and B fields are indicated by the T character, which may vary in value from 0 through 255, with 0 indicating a length of 256 bytes.

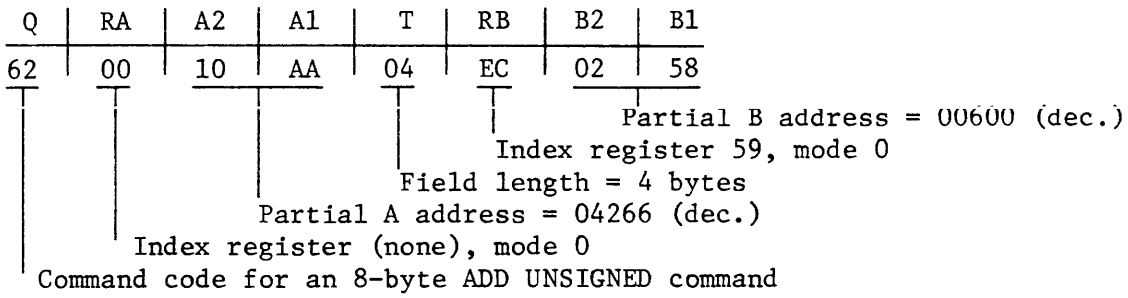
CAUTION
Do not add hexadecimal fields; the result is predictable, but not a correct hexadecimal sum.

Addition is performed from right to left, one byte at a time, using only the rightmost four bits of each byte; the leftmost four bits are ignored. A carry beyond the rightmost four bits of one byte is added to the byte on its left. A carry beyond the leftmost byte is discarded after setting the overflow flag ON; if a carry is not generated beyond the leftmost byte, the initial state of the overflow flag is preserved. The result of the addition is stored in the rightmost four bits of each B field byte; the ASCII bit configuration 0011 is stored in the leftmost four bits.

The result obtained from adding the contents of overlapping A and B fields may differ from the result obtained when adding the contents of A and B fields that do not overlap (assuming the A and B values are the same in both instances). The addition of overlapping fields normally changes the contents of the A field.

The implied B address for a subsequent 4-byte command is the same as the effective B address established prior to execution of the ADD UNSIGNED command.

Example



Assume:

Index register 59 contains 104C (05132 dec.)

After command setup:

Effective A address = 10AA (04266 dec.)
Effective B address = 0258 + 140C = 1664 (05732 dec.)

Before command execution:

A field address (hex.)	10AA	10AB	10AC	10AD
A field contents (ASCII)	0011 0100	0011 0111	0011 0010	0011 1000
Decimal equivalent	4	7	2	8

B field address (hex.)	1664	1665	1666	1667
B field contents (ASCII)	0011 0110	0011 0100	0011 0100	0011 0011
Decimal equivalent	6	4	4	3

Arithmetic manipulation:

Decimal equivalent of addition to be performed:

```
  4728
  6443
 11171
```

ASCII addition:

1. Excess six is added to b4 - b1 of each A field character.

A field characters (b4 - b1)	0100	0111	0010	1000
Excess six	0110	0110	0110	0110
	0010	0001	0100	1110
Carries	1	11	11	
Result	1010	1101	1000	1110

2. B4 - b1 of each corresponding B field character are added to the result obtained in step 1.

Results of step 1	1010	1101	1000	1110
B field characters (b4 - b1)	0110	0100	0100	0011
	1100	1001	1100	1101
Carries	1 11 1	1	1	11
Uncorrected result	0001	0001	1101	0001

Carry sets the hardware overflow flag ON.

3. A binary ten is added to the uncorrected result of any digit addition from step 2 that did not generate a carry to the digit on its left (including the leftmost digit).

Uncorrected result	0001	0001	1101	0001	
Binary ten correction			1010		
	0001	0001	0111	0001	
Carry		1			← Carries between
Final result	0001	0001	0111	0001	digits are ignored.

4. The final result is stored in b4 - b1 of the corresponding B field characters, with the zone bits 0011 stored in b8 - b5 of each character.

Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	1664	1665	1666	1667
B field contents (ASCII)	0011 0001	0011 0001	0011 0111	0011 0001
Decimal equivalent	1	1	7	1

The implied B address for a 4-byte command following the execution of this ADD UNSIGNED command is 1664 (05732 dec.).

BRANCH EQUAL (BRE)

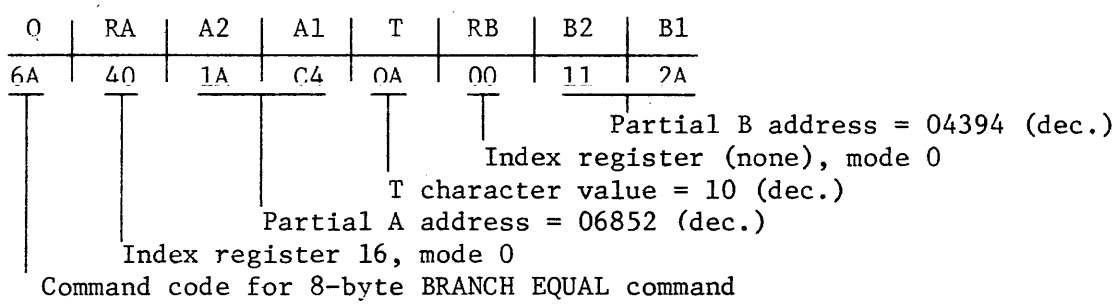
Command code: 106 (6A) (EA)

The BRANCH EQUAL command tests the E flag to determine whether to transfer program control to the next command in sequence or to transfer control to the command indicated by the effective A address. If the E flag is ON, control is transferred to the command indicated by the effective A address; no return link is established. If the E flag is OFF, program control is transferred to the next command in sequence. In either instance, the repeat indicator (if ON) is set OFF.

The effective A address must be 0 mod 4 (evenly divisible by four); otherwise, a program error occurs and the address of the next command in sequence is left in the control register.

The T character and effective B address of the 8-byte command are not used during execution; however, they are set up and available as implied operands for the next command to be executed. The implied B address is the same as the effective B address established prior to execution; it is available to a subsequent 4-byte command, whether that command is the next in sequence or the result of a branch.

Example



Assume:

Index register 16 contains 01A0 (00416 dec.)
The E flag is ON.

After command setup:

Effective A address = 1AC4 + 01A0 = 1C64 (07268 dec.)
Effective B address = 112A (04394 dec.)

Following command execution:

Program control is transferred to the command at address 1C64.

The implied B address for a 4-byte command following the execution of this BRANCH EQUAL command is 112A (04394 dec.).

BRANCH GREATER (BRG)

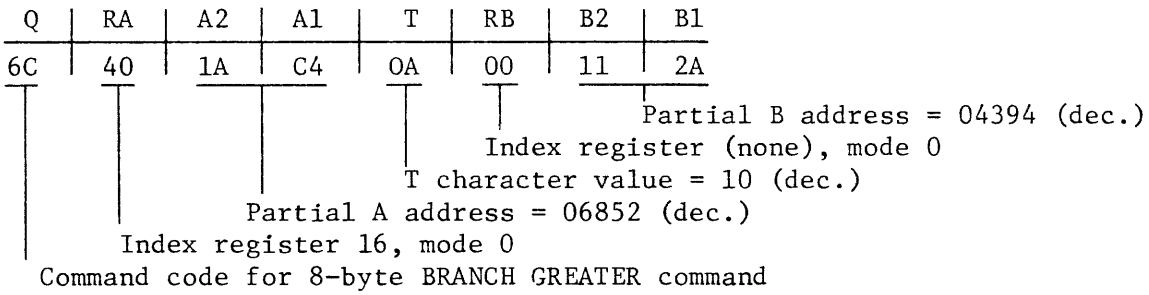
Command code: 108 (6C) (EC)

The BRANCH GREATER command tests the G flag to determine whether to transfer program control to the next command in sequence or to transfer control to the command indicated by the effective A address. If the G flag is ON, control is transferred to the command indicated by the effective A address; no return link is established. If the G flag is OFF, program control is transferred to the next command in sequence. In either instance, the repeat indicator (if ON) is set OFF.

The effective A address must be 0 mod 4 (evenly divisible by four); otherwise, a program error occurs and the address of the next command in sequence is left in the control register.

The T character and effective B address of the 8-byte command are not used during execution; however, they are set up and available as implied operands for the next command to be executed. The implied B address is the same as the effective B address established prior to execution; it is available to a subsequent 4-byte command, whether that command is the next in sequence or the result of a branch.

Example



Assume:

Index register 16 contains 01A0 (00416 dec.)
The G flag is ON.

After command setup:

Effective A address = 1AC4 + 01A0 = 1C64 (07268 dec.)
Effective B address = 112A (04394 dec.)

Following command execution:

Program control is transferred to the command at address 1C64.

The implied B address for a 4-byte command following the execution of this BRANCH GREATER command is 112A (04394 dec.)

BRANCH GREATER OR EQUAL (BRGE)

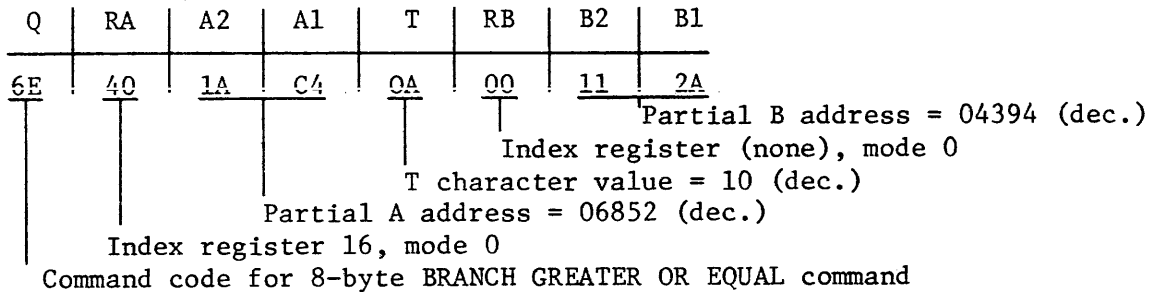
Command code: 110 (6E) (EE)

The BRANCH GREATER OR EQUAL command tests both the G and E flags to determine whether to transfer program control to the next command in sequence or to transfer control to the command indicated by the effective A address. If either the G or E flag is ON, control is transferred to the command indicated by the effective A address; no return link is established. If both the G and E flags are OFF, program control is transferred to the next command in sequence. In either instance, the repeat indicator (if ON) is set OFF.

The effective A address must be 0 mod 4 (evenly divisible by four); otherwise, a program error occurs and the address of the next command in sequence is left in the control register.

The T character and effective B address of the 8-byte command are not used during execution; however, they are set up and available as implied operands for the next command to be executed. The implied B address is the same as the effective B address established prior to execution; it is available to a subsequent 4-byte command, whether that command is the next in sequence or the result of a branch.

Example



Assume:

Index register 16 contains 01A0 (00416 dec.)
The G flag is ON.

After command setup:

Effective A address = 1AC4 + 01A0 = 1C64 (07268 dec.)
Effective B address = 112A (04394 dec.)

Following command execution:

Program control is transferred to the command at address 1C64.

The implied B address for a 4-byte command following the execution of this BRANCH GREATER OR EQUAL command is 112A (04394 dec.).

BRANCH LESS (BRL)

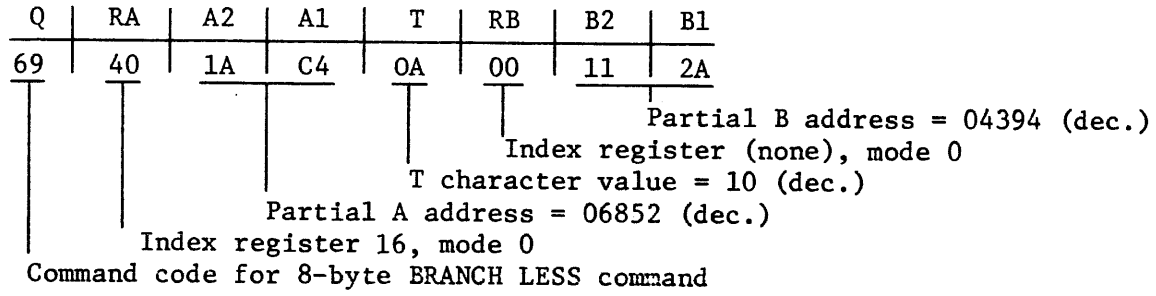
Command code: 105 (69) (E9)

The BRANCH LESS command tests the L flag to determine whether to transfer program control to the next command in sequence or to transfer control to the command indicated by the effective A address. If the L flag is ON, control is transferred to the command indicated by the effective A address; no return link is established. If the L flag is OFF, program control is transferred to the next command in sequence. In either instance, the repeat indicator (if ON) is set OFF.

The effective A address must be 0 mod 4 (evenly divisible by four); otherwise, a program error occurs and the address of the next command in sequence is left in the control register.

The T character and effective B address of an 8-byte command are not used during execution; however, they are set up and available as implied operands for the next command to be executed. The implied B address is the same as the effective B address established prior to execution; it is available to a subsequent 4-byte command, whether that command is the next in sequence or the result of a branch.

Example



Assume:

Index register 16 contains 01A0 (00416 dec.)
The L flag is ON.

After command setup:

Effective A address = 1AC4 + 01A0 = 1C64 (07268 dec.)
Effective B address = 112A (04394 dec.)

Following command execution:

Program control is transferred to the command at address 1C64.

The implied B address for a 4-byte command following the execution of this BRANCH LESS command is 112A (04394 dec.).

BRANCH LESS OR EQUAL (BRLE)

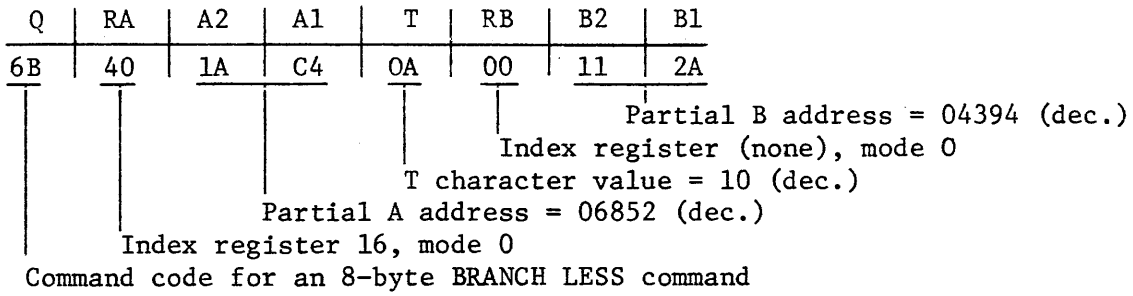
Command code: 107 (6B) (EB)

The BRANCH LESS OR EQUAL command tests both the L and E flags to determine whether to transfer program control to the next command in sequence or to transfer control to the command indicated by the effective A address. If either the L or E flag is ON, control is transferred to the command indicated by the effective A address; no return link is established. If both the L and E flags are OFF, program control is transferred to the next command in sequence. In either instance, the repeat indicator (if ON) is set OFF.

The effective A address must be 0 mod 4 (evenly divisible by four); otherwise, a program error occurs and the address of the next command in sequence is left in the control register.

The T character and effective B address of an 8-byte command are not used during execution; however, they are set up and available as implied operands for the next command to be executed. The implied B address is the same as the effective B address established prior to execution; it is available to a subsequent 4-byte command, whether that command is the next in sequence or the result of a branch.

Example



Assume:

Index register 16 contains 01A0 (00416 dec.)
The E flag is ON.

After command setup:

Effective A address = 1AC4 + 01A0 = 1C64 (07268 dec.)
Effective B address = 112A (04394 dec.)

Following command execution:

Program control is transferred to the command at address 1C64.

The implied B address for a 4-byte command following the execution of this BRANCH LESS OR EQUAL command is 112A (04394 dec.).

BRANCH LESS OR GREATER (BRU)

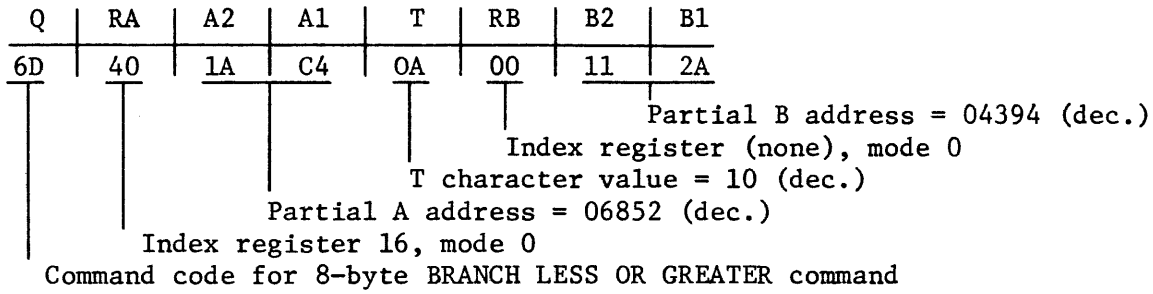
Command code: 109 (6D) (ED)

The BRANCH LESS OR GREATER command tests both the L and G flags to determine whether to transfer program control to the next command in sequence or to transfer control to the command indicated by the effective A address. If either the L or G flag is ON, control is transferred to the command indicated by the effective A address; no return link is established. If both the L and G flags are OFF, program control is transferred to the next command in sequence. In either instance, the repeat indicator (if ON) is set OFF.

The effective A address must be 0 mod 4 (evenly divisible by four); otherwise, a program error occurs and the address of the next command in sequence is left in the control register.

The T character and effective B address of an 8-byte command are not used during execution; however, they are set up and available as implied operands for the next command to be executed. The implied B address is the same as the effective B address established prior to execution; it is available to a subsequent 4-byte command, whether that command is the next in sequence or the result of a branch.

Example



Assume:

Index register 16 contains 01A0 (00416 dec.)
The G flag is ON.

After command Setup:

Effective A address = 1AC4 + 01A0 = 1C64 (07268 dec.)
Effective B address = 112A (04394 dec.)

Following command execution:

Program control is transferred to the command at address 1C64.

The implied B address for a 4-byte command following the execution of this BRANCH LESS OR GREATER command is 112A (04394 dec.).

BRANCH OVERFLOW (BROV)

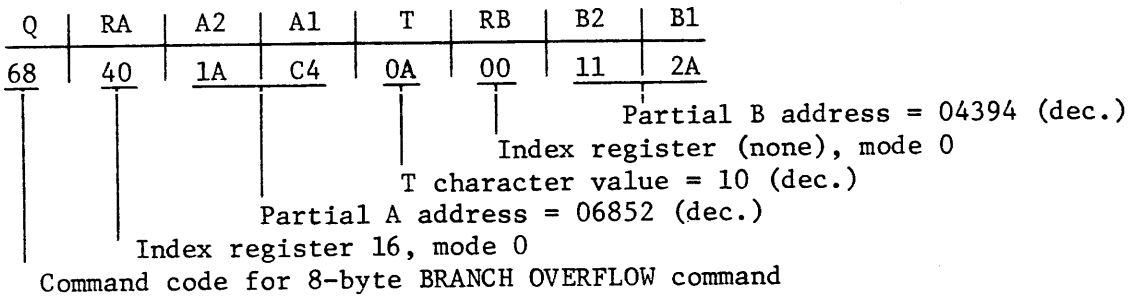
Command code: 104 (68) (E8)

The BRANCH OVERFLOW command tests the overflow flag to determine whether to transfer program control to the next command in sequence or to transfer control to the command indicated by the effective A address. If the overflow flag is ON, control is transferred to the command indicated by the effective A address; no return link is established and the overflow flag is set OFF. If the overflow flag is OFF initially, program control is transferred to the next command in sequence. In either instance, the repeat indicator (if ON) is set OFF.

The effective A address must be 0 mod 4 (evenly divisible by four); otherwise, a program error occurs and the address of the next command in sequence is left in the control register.

The T character and effective B address of an 8-byte command are not used during execution; however, they are set up and available as implied operands for the next command to be executed. The implied B address is the same as the effective B address established prior to execution; it is available to a subsequent 4-byte command, whether that command is the next in sequnece or the result of a branch.

Example



Assume:

Index register 16 contains 01A0 (00416 dec.)
The overflow flag is ON.

After command setup:

Effective A address = 1AC4 + 01A0 = 1C64 (07268 dec.)
Effective B address = 112A (04394 dec.)

Following command execution:

Program control is transferred to the command at address 1C64.

The implied B address for a 4-byte command following the execution of this BRANCH OVERFLOW command is 112A (04394 dec.).

BRANCH UNCONDITIONALLY (BR)

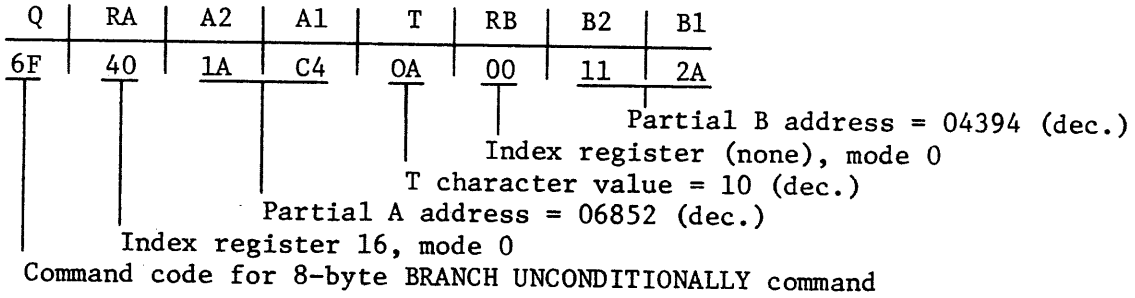
Command code: 111 (6F) (EF)

The BRANCH UNCONDITIONALLY command transfers program control to the command indicated by the effective A address; no return link is established and the repeat indicator (if ON) is set OFF.

The effective A address must be 0 mod 4 (evenly divisible by four); otherwise, a program error occurs and the address of the next command in sequence is left in the control register.

The T character and effective B address of an 8-byte command are not used during execution; however, they are set up and available as implied operands for a subsequent 4-byte command. The implied B address is the same as the effective B address established prior to the execution of the BRANCH UNCONDITIONALLY command.

Example



Assume:

Index register 16 contains 01A0 (00416 dec.)

After command setup:

Effective A address = 1AC4 + 01A0 = 1C64 (07268 dec.)
Effective B address = 112A (04394 dec.)

Following command execution:

Program control is transferred to the command at address 1C64.

The implied B address for a 4-byte command following the execution of this BRANCH UNCONDITIONALLY command is 112A (04394 dec.).

COMPARE BINARY (BCOMP)

Command code: 101 (65) (E5)

The COMPARE BINARY command compares the binary value of the contents of the field specified by the effective A address to the binary value of the field specified by the effective B address; the appropriate flag (L, E, or G) is set ON to indicate the result. The lengths of both the A and B fields are indicated by the T character which may vary in value from 0 through 255, with 0 indicating a length of 256 bytes.

The binary comparison is made on the entire field from right to left, one byte at a time with the following results:

1. If the A field value is less than the B field value, the L flag is set ON; the E and G flags are set OFF. The initial state of the repeat indicator is retained.
2. If the A field value is equal to the B field value, the E flag is set ON; the L flag, G flag, and repeat indicator are set OFF.
3. If the A field value is greater than the B field value, the G flag is set ON; the L flag, E flag, and the repeat indicator are set OFF.

In all instances, following command execution, the initial contents of the A and B fields are unchanged. The implied B address for a subsequent 4-byte command is the same as the effective B address established prior to the execution of the COMPARE BINARY command.

Effectively, the COMPARE BINARY command uses complementary addition to determine the result of a comparison; the result and the method used to obtain it is shown in the following examples.

Example 1 (A field value less than B field value)

Q	RA	A2	A1	T	RB	B2	B1
65	00	10	F0	03	00	11	0A

Partial B address = 04262 (dec.)
Index register (none), mode 0
Field length = 3 bytes
Partial A address = 04336 (dec.)
Index register (none), mode 0
Command code for an 8-byte COMPARE BINARY command

After command setup:

Effective A address = 10F0 (04336 dec.)
Effective B address = 110A (04262 dec.)

Before command execution:

A field address (hex.)	10F0	10F1	10F2
A field contents (ASCII)	0011 0111	0011 0110	0011 1001
Decimal equivalent	7	6	9

B field address (hex.)	110A	110B	110C
B field contents (ASCII)	0011 1001	0011 0100	0011 1001
Decimal equivalent	9	4	9

Binary comparison:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost byte of the A fields 1's complement.

1's complement	11001000	11001001	11000110
Initial carry			1
2's complement	11001000	11001001	11000111

2. The contents of the B field are added to the A fields 2's complement.

2's complement	11001000	11001001	11000111
B field contents	00111001	00110100	00111001
	11110001	11111101	11111110
Carries	1 1111	11	1111111
Final result	00000001	11111110	00000000

3. Because the result is not equal to zero and there is a carry beyond the leftmost byte, the L flag is set ON.

Following command execution:

The A and B field contents are unchanged, and the L flag is ON.

The implied B address for a 4-byte command following the execution of this COMPARE BINARY command is 110A (04362 dec.).

Example 2 (A field value equal to B field value)

Assume that the fields used in the preceding example contain the following information before command execution:

A field address (hex.)	10F0	10F1	10F2
A field contents (ASCII)	0100 0001	0011 1101	0100 0010
Equivalent	A	=	B

B field address (hex.)	110A	110B	110C
B field contents (ASCII)	0100 0001	0011 1101	0100 0010
Equivalent	A	=	B

Binary comparison:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost byte of the A field 1's complement.

1's complement	10111110	11000010	10111101
Initial carry			1
	10111110	11000010	10111100
Carry			1
2's complement	10111110	11000010	10111110

2. The contents of the B field are added to the 2's complement of the A field.

2's complement	10111110	11000010	10111110
B field contents	01000001	00111101	01000010
	11111111	11111111	11111100
Carries	1 11111111	11111111	111111
Final result	00000000	00000000	00000000

3. Because the result is equal to zero and there is a carry beyond the leftmost byte, the E flag is set ON.

Following command execution:

The A and B field contents are unchanged, the E flag is ON, and the repeat indicator is OFF.

The implied B address for a 4-byte command following the execution of this COMPARE BINARY command is 110A (04262 dec.).

Example 3 (A field value greater than B field value)

Assume that the fields defined in example 1 contain the following information before command execution:

A field address (hex.)	10F0	10F1	10F2
A field contents (ASCII)	0100 1110	0100 0011	0101 0010
Equivalent	N	C	R

B field address (hex.)	110A	110B	110C
B field contents (ASCII)	0100 1001	0100 0011	0100 1101
Equivalent	I	C	M

Binary comparison:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost byte of the A fields 1's complement.

1's complement	10110001	10111100	10101101
Initial carry			1
	10110001	10111100	10101100
Carry			1
2's complement	10110001	10111100	10101110

2. The contents of the B field are added to the 2's complement of the A field.

2's complement	10110001	10111100	10101110
B field contents	01001001	01000011	01001101
	11111000	11111111	11100011
Carries	1		11
Final result	11111010	11111111	11111011

3. Because there is not a carry generated beyond the leftmost byte of the result, the G flag is set ON.

Following command execution:

The A and B field contents are unchanged, the G flag is ON, and the repeat indicator is OFF.

The implied B address for a 4-byte command, following the execution of this COMPARE BINARY command, is 110A (04262 dec.).

COMPARE SIGNED (PCOMP)

Command code: 69 (45) (C5)

The COMPARE SIGNED command algebraically compares the decimal value of the contents of the field specified by the effective A address to the decimal value of the field specified by the effective B address; the appropriate flag (L, E, or G) is set ON to indicate the result. The initial contents of the A and B fields are considered to be signed, packed (see PACK command description) decimal information, with the sign stored in the rightmost four bits of each field. Fields with negative values are indicated by the bit configuration 1101; any other configuration indicates the field to contain a positive value. The lengths of both the A and B fields are indicated by the T character, which may vary in value from 0 through 255, with 0 indicating a length of 256 bytes.

CAUTION
Do not compare hexadecimal fields; the result is predictable, but not a valid comparison.

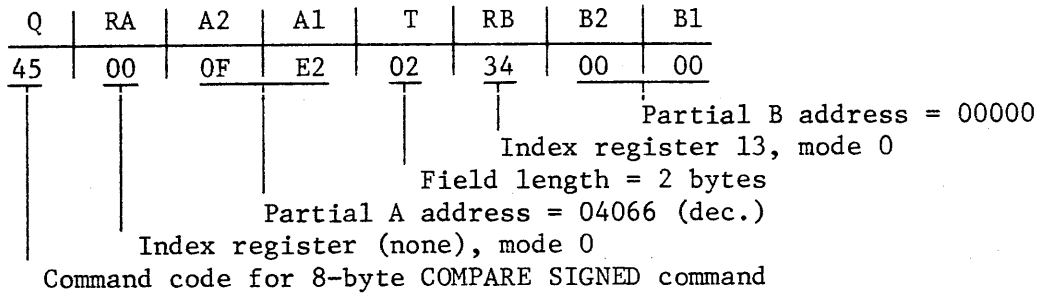
The algebraic comparison is made on the entire field from right to left, one byte at a time with the following results:

1. If the A field algebraic value is less than the B field algebraic value, the L flag is set ON; the E and G flags are set OFF. The initial state of the repeat indicator is retained.
2. If the A field algebraic value is equal to the B field algebraic value, the E flag is set ON; the L flag, G flag and repeat indicator are set OFF. (If the digits of both fields are zero, regardless of the signs, the E flag is set ON.)
3. If the A field algebraic value is greater than the B field algebraic value, the G flag is set ON; the L flag, E flag, and repeat indicator are set OFF.

In all cases, following command execution, the initial contents of the A and B fields are unchanged. The implied B address for a subsequent 4-byte command is the same as the effective B address established for the COMPARE SIGNED command prior to execution.

Effectively, the COMPARE SIGNED command uses the signs of the fields and complementary addition to determine the result of a comparison. The result and the method used to obtain it are shown in the following examples.

Example 1 (A field value greater than B field value)



Assume:

Index register 13 contains 10A8 (04264 dec.)

After command setup:

Effective A address = 0FE2 (04066 dec.)
Effective B address = 0000 + 10A8 = 10A8 (04264 dec.)

Before command execution:

A field address (hex.)	OFE2		OFE3	
A field contents (BCD)	0110	1000	0011	1101
Decimal equivalent	6	8	3	-

B field address (hex.)	10A8		10A9	
B field contents (BCD)	0111	0100	0010	1101
Decimal equivalent	7	4	2	-

Comparison:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost digit (not the sign) of the A fields 1's complement.

1's complement	1001	0111	1100
Initial carry			1
2's complement	1001	0111	1101

2. The contents of the B field (not including the sign) are added to the 2's complement of the A field.

2's complement	1001	0111	1101
B field contents	0111	0100	0010
	1110	0011	1111
Carries	1 111	1	
Final result	0000	1011	1111

3. The carry beyond the leftmost byte of the result, the final result not being equal to zero, and the fact that both the A and B fields have negative signs, sets the G flag ON.

Following command execution:

The contents of the A and B fields are unchanged, the G flag is ON, and the repeat indicator is OFF.

The implied B address for a 4-byte command following the execution of this COMPARE SIGNED command is 10A8 (04264 dec.).

Example 2 (A field value equal to B field value)

Assume that the fields used in the preceding example contain the following information before command execution:

A field address (hex.)	OFE2		OFE3	
A field contents (BCD)	0111	0010	0101	1011
Decimal equivalent	7	2	5	+

B field address (hex.)	10A8		10A9	
B field contents (BCD)	0111	0010	0101	1011
Decimal equivalent	7	2	5	+

Comparison:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost digit (not the sign) of the A fields 1's complement.

1's complement	1000	1101	1010
Initial carry			1
2's complement	1000	1101	1011

2. The contents of the B field (not including the sign) are added to the 2's complement of the A field.

2's complement		1000	1101	1011
B field contents		0111	0010	0101
		1111	1111	1110
Carries	1	1111	1111	111
Final result		0000	0000	0000

3. The carry beyond the leftmost byte of the result, the result being equal to zero, and the fact that both fields have like signs, sets the E flag ON.

Following command execution:

The A and B field contents are unchanged, the E flag is ON, and the repeat indicator is OFF.

The implied B address for a 4-byte command following the execution of this COMPARE SIGNED command is 10A8 (04264 dec.).

Example 3 (A field value less than B field value)

Assume that the fields used in example 1 contain the following information before command execution:

A field address (hex.)	OFE2		OFE3	
A field contents (BCD)	0001	0010	0000	1101
Decimal equivalent	1	2	0	-

B field address (hex.)	10A8		10A9	
B field contents (BCD)	0000	0001	0101	1011
Decimal equivalent	0	1	5	+

Comparison:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost digit (not the sign) of the A fields 1's complement.

1's complement	1110	1101	1111
Initial carry			1
	1110	1101	1110
Carries		11	111
2's complement	1110	1110	0000

2. The B field contents (not including the sign) are added to the 2's complement of the A field.

2's complement	1110	1110	0000
B field contents	0000	0001	0101
Final result	1110	1111	0101

3. Since there is not a carry generated beyond the leftmost byte of the result, the result is not equal to zero, and the signs have been compared, the L flag is set ON.

Following command execution:

The contents of the A and B fields are unchanged, the L flag is ON, and the initial state of the repeat indicator is preserved.

The implied B address for a 4-byte command following the execution of this COMPARE SIGNED command is 10A8 (04264 dec.).

COUNT (COUNT)

Command Code: 74 (4A) (CA)

The COUNT command obtains the content of the 1-character binary field at memory location 64 (called the COUNT counter), decrements that value by 1, and restores it to memory. The command then compares the resulting value to zero (zero decremented by 1 is 255). If the result is not equal to zero, the COUNT command transfers program control to the command indicated by the effective A address. If the result is equal to zero, the next command in sequence is executed. In either case, the repeat indicator (RI) is turned OFF. If the effective A address is not a legal command address, a program error occurs, leaving the control register undisturbed.

The B operand and the T character are not used.

DECODE ALL (DCODA)

Command code: 79 (4F) (CF)

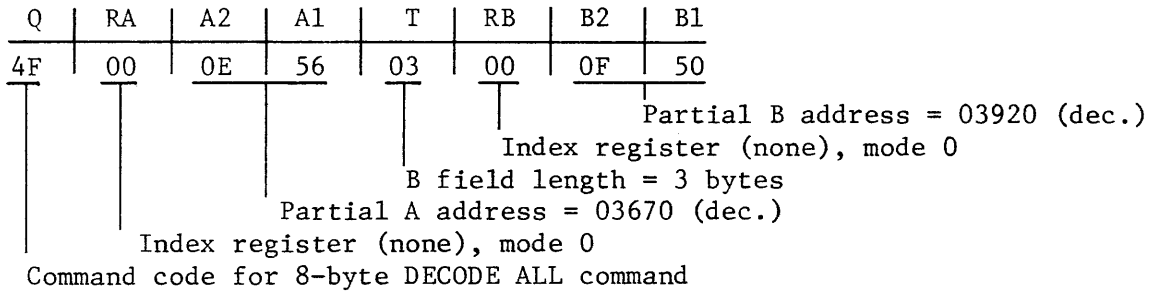
The DECODE ALL command replaces each character of the field specified by the effective B address with a corresponding character from a table specified by the effective A address. This decoding process is repeated for each character in the B field, beginning with the leftmost byte. The T character value,

which may vary from 0 through 255 (with 0 indicating 256), indicates the length of the B field in bytes.

The address of each replacement character is computed by adding the binary value of the B field character to the effective A address; the replacement character then replaces the original B field character. When the rightmost B field character has been decoded, the G flag is set ON and the command terminates.

The implied B address for a subsequent 4-byte command is equal to the sum of adding the length of the B field to the effective B address; it is placed into both the appropriate hardware register and index register nine.

Example



After command setup:

Effective A address = 0E56 (03670 dec.)
Effective B address = 0F50 (03920 dec.)

Before command execution:

Table address (hex.)	0E56	0E57	0E58	0E59
Table contents	0100 0001	0100 0010	0100 0011	0100 0100
Equivalent (ASCII)	A	B	C	D

B field address (hex.)	0F50		0F51		0F52	
B field contents	0000	0010	0000	0011	0000	0000
Hexadecimal equivalent	0	2	0	3	0	0

During command execution:

The replacement character address is computed by adding the contents of the B field address to the effective A address; the replacement character at the resulting new address replaces the original contents of the B field address.

1a. B field address 0F50
Original contents 0000 0010
Equivalent 0 2

1b. Effective A address 0E56
Original B contents 02
Replacement character address 0E58

1c. B field address 0F50
New contents 0100 0011
Equivalent C

2a. B field address OF51
Original contents 0000 0011
Equivalent 0 3

2b. Effective A address OE56
Original B contents 03
Replacement character address OE59

2c. B field address OF51
New contents 0100 0100
Equivalent D

3a. B field address OF52
Original contents 0000 0000
Equivalent 0 0

3b. Effective A address OE56
Original B contents 00
Replacement character address OE56

3c. B field address OF52
New contents 0100 0001
Equivalent A

Following command execution:

The contents of the table are unchanged and the G flag is ON.

B field address (hex.)	OF50	OF51	OF52
B field contents	0100 0011	0100 0100	0100 0001
Equivalent (ASCII)	C	D	A

The implied B address for a 4-byte command following the execution of this DECODE ALL command is OF53 (03923 dec.), and is contained in the appropriate hardware register and in index register nine.

DECODE TO DELIMITER (DCODD)

Command code: 78 (4E) (CE)

The DECODE TO DELIMITER command replaces each character of the field specified by the effective B address with a corresponding character from a table specified by the effective A address. This decoding process is repeated for each character in the B field, beginning with the leftmost byte, until either all B field characters are decoded or a delimiter character is detected in the table. The T character, which may vary in value from 0 through 255 (with 0 indicating 256), specifies the length of the B field in bytes.

The address of each replacement character is computed by adding the binary value of the B field character to the effective A address; the replacement character which is read from memory, replaces the original B field character.

Termination of the command occurs on either of two conditions with the following results:

1. If the rightmost character of the B field has been decoded, the G flag is set ON (providing the delimiter character is not encountered) and execution terminates.
2. If a delimiter character (a byte with bit eight ON) is encountered during character replacement, the E flag is set ON and execution terminates; the delimiter character is not stored in the B field.

The implied B address for a subsequent 4-byte command varies depending on the cause of command termination. If all the characters of the B field are decoded the resultant implied B address is equal to the sum of adding the length of the B field to the effective B address. If a delimiter character terminates execution the implied B address is pointing one byte beyond the last B field characte that was replaced. In either case, the address is stored in both the appropriat hardware register and in index register nine.

Example

Q	RA	A2	A1	T	RB	B2	B1
4E	00	0F	FD	03	00	10	A0

Partial B address = 04256 (dec.)
Index register (none), mode 0
B field length = 3 bytes
Partial A address = 04093 (dec.)
Index register (none), mode 0
Command code for 8-byte DECODE TO DELIMITER command

After command setup:

Effective A address = 0FFD (04093 dec.)
Effective B address = 10A0 (04256 dec.)

Before command execution:

Table address (hex.)	OFFD	OFFE	OFFF	1000
Table contents	0100 0001	0100 0010	0100 0011	1000 0000
Equivalent	A	B	C	Delimiter Character

B field address (hex.)	10A0		10A1		10A2	
B field contents	0000	0010	0000	0000	0000	0011
Hexadecimal equivalent	0	2	0	0	0	3

During command execution:

The replacement character address is computed by adding the contents of the B field address to the effective A address; the replacement character at the resulting new address replaces the original contents of the B field address.

1a. B field address — 10A0
Original contents — 0000 0010
Equivalent — 0 2

1b. Effective A address — OFFD
Original B contents — 02
Replacement character address — OFFF

1c. B field address — 10A0
New contents — 0100 0011
Equivalent — C

2a. B field address — 10A1
Original contents — 0000 0000
Equivalent — 0 0

2b. Effective A address — OFFD
Original B contents — 00
Replacement character address — OFFD

2c. B field address 10A1
New contents 0100 0001
Equivalent A

3a. B field address 10A2
Original contents 0000 0011
Equivalent 0 3

3b. Effective A address OFFD
Original B contents 03
Replacement character address 1000

3c. B field address 10A2
New contents 0000 0011
Equivalent 0 3

The replacement character for address 10A2 is the delimiter character; the command terminates and the original contents of this address are retained.

Following command execution:

The contents of the table are unchanged and the E flag is ON.

B field address (hex.)	10A0	10A1	10A2
B field contents	0100 0011	0100 0001	0000 0011
Equivalent	C	A	0 3

New contents Original contents

The implied B address for a 4-byte command following the execution of this DECODE TO DELIMITER command is 10A2 (04258 dec.), and is contained in both the appropriate hardware register and in index register nine.

DIVIDE (PDIV)

Command code: 113 (71) (F1)

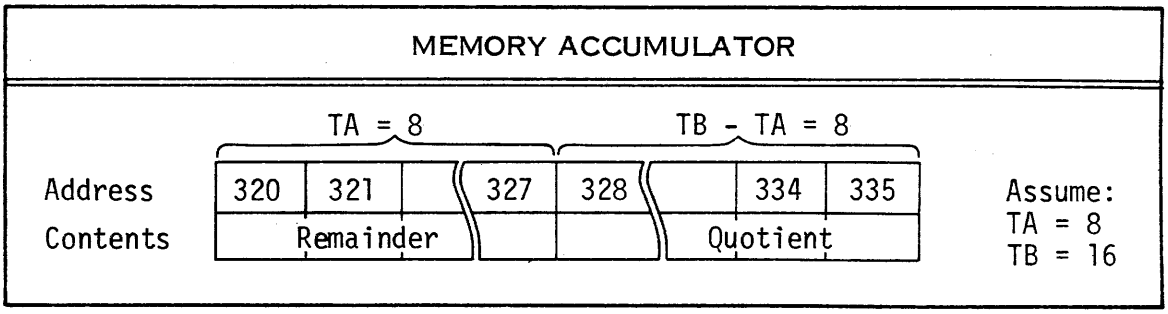
The DIVIDE command, which is optional with the NCR Century 101 Processor, divides the contents of the field specified by the effective B address (dividend) by the contents of the field specified by the effective A address (divisor), and stores the result in the memory accumulator.

In the command structure, the effective A address must be 0 mod 4 (evenly divisible by 4), or a program error occurs. The contents of the A and B fields are assumed to be signed and packed (see PACK command description), with the signs stored in the rightmost four bits of each field, decimal points are assumed to the right of the least significant digit.

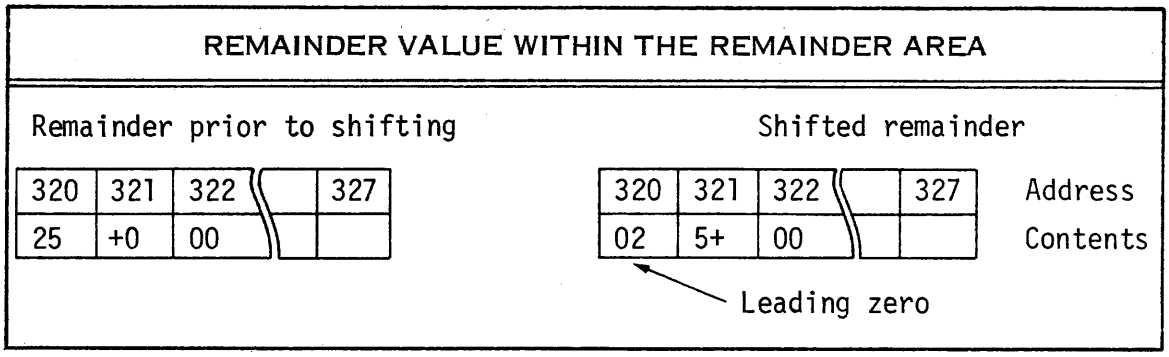
The leftmost four bits of the T character (TA) indicate the length of the A field and may vary in value from 0 through 7, with 0 indicating eight bytes (a program error occurs if the leftmost bit of TA is ON). The rightmost four bits of the T character (TB) indicate the length of the B field and may vary in value from 0 through 15, with 0 indicating a length of 16 bytes. TA and TB are used to determine whether the quotient is legal in size. If the result of subtracting TA (the length of the divisor) from TB (the length of the dividend) is greater than eight, 0 or negative, a program error occurs.

The result of the division (both quotient and remainder) is stored as signed, packed data in the memory accumulator, which is a fixed area in memory beginning at decimal address 320 and ending at address 335. The sign of the quotient is determined according to algebraic laws; the remainder assumes a like sign.

The quotient area is eight bytes in length, beginning at memory location 328 and terminating at 335; the quotient is stored in this area, right-justified and zero-filled to the left, and occupies the number of bytes indicated by TB - TA (all bytes to the left of the quotient in the quotient area are unchanged). The remainder area is also eight bytes in length, beginning at location 320 (decimal). The remainder is stored in this area (left-justified and zero-filled to the right) and occupies the number of bytes indicated by TA (the length of the divisor); all bytes to the right of the remainder area including location 327 are unchanged.



If the sign of the remainder is located in the leftmost four bits of a byte, the entire remainder value is shifted to the right one digit position within the remainder field, and a leading zero is added.



Before any division occurs, a check is made to determine whether the quotient will fit within the allotted number of bytes (TB - TA) of the memory accumulator. The check is performed in three steps:

1. The divisor is modified by placing a zero to the left of the leftmost digit and deleting the sign.

2. The dividend is modified by deleting TB - TA bytes (subtracting the value of the TA character from the TB character value), beginning with the rightmost byte.
3. The modified divisor is subtracted from the modified dividend.

If the result of the subtraction is negative (indicating that the quotient fits within the allotted number of bytes in the memory accumulator), the division process takes place; otherwise, a program error occurs, leaving the initial contents of the memory accumulator undisturbed.

Division is achieved by a trial quotient method similar to that used in manual division. For example, the logic circuitry compares the leftmost non-zero digit of the divisor to the leftmost non-zero digit of the dividend.

```
04230 ⌈00006791
```

If the divisor digit is equal to or less than the dividend digit, the logic circuitry assigns the appropriate trial quotient (4 goes into 6 once). The entire divisor value is then multiplied by the quotient value, and the resulting product is subtracted from the dividend.

```
                1  True quotient
04230 ⌈00006791
       4230
       2561  Remainder
```

If the divisor digit is larger than the dividend digit (04230 ⌈00326047), the logic circuitry assumes a dividend digit equal to 10 times its original value, plus nine (30 + 9 = 39). The trial quotient is then computed on the basis of the assumed dividend value (4 goes into 39 nine times). The entire divisor value is then multiplied by the trial quotient value, and the resulting product is subtracted from the dividend. If the subtraction results in a negative value, the trial quotient value is too large; if it results in a positive value, the trail quotient value becomes the true quotient value.

```
                9
04230 ⌈00326047
       38070
       9994534
```

When the trial quotient value is too large, the processor decrements the trial quotient by one and adds the entire divisor value to the negative result. This process is repeated until the remainder becomes positive again; at that time, the trial quotient value becomes the true quotient value.

```
                 7 - True quotient value
                 8 - Trial quotient value
                 9 - Trial quotient value
04230  |00326047
          38070
        9994534  Negative value (trial quotient of 9)
           4230
        9998764  Negative value (trial quotient of 8)
           4230
        0002994  Positive value (trial quotient of 7)
```

After the first true value has been determined, the processor repeats the process, using the first non-zero digit of the divisor and the first non-zero digit of the positive value, plus the next digit of the dividend.

```
                 7
                 8
                 9
04230  |00326047
          38070
        9994534
           4230
        9998764
           4230
        00029947
```

Following the execution of the divide command, the implied B address available for a subsequent 4-byte command is 0140 (00320 dec.).

CAUTION
Memory locations 288 through 319 (decimal) are used by the DIVIDE command to store intermediate results and should not be used for other purposes.

Example

Q	RA	A2	A1	T	RB	B2	B1
71	74	01	F0	35	7C	04	00

Partial B address = 01024 (dec.)
Index register 31, mode 0
A field length (TA) = 3 bytes; B field length (TB) = 5 bytes
Partial A address = 00496 (dec.)
Index register 29, mode 0
Command code for 8-byte DIVIDE command

Assume:

Index register 29 contains 0B54
Index register 31 contains 0C1C

After command setup:

Effective A address = 01F0 + 0B54 = 0D44 (03396 dec.)
Effective B address = 0400 + 0C1C = 101C (04124 dec.)

Before command execution:

A field address (hex.)	0D44		0D45		0D46	
A field contents (BCD)	0000	0100	0010	0011	0000	1101
Decimal equivalent	0	4	2	3	0	-

B field address (hex.)	101C		101D		101E		101F		1020	
B field contents (BCD)	0000	0000	0000	0000	0110	0111	0110	1001	0001	1011
Decimal equivalent	0	0	0	0	6	7	6	9	1	+

Accumulator

Address (hex.)	0140		0141		014E		014F	
Contents	1111	1111	1111	1111	1111	1111	1111	1111
Equivalent (hex.)	F	F	F	F	F	F	F	F

Arithmetic manipulation:

Decimal equivalent of division to be performed: $-4230 \overline{)+67691}$ $\begin{matrix} -16 \\ \end{matrix}$ r = -11

Quotient overflow check:

1. The divisor is modified by deleting the sign and adding a leading zero.

   Original divisor = 04230-

   Modified divisor = 004230

2. The dividend is modified by subtracting the TA character value from the TB character value and deleting TB - TA bytes, starting with the rightmost byte.

   Original dividend = 000067691+

   Number of bytes to be deleted = TB - TA = 5 - 3 = 2

   Modified dividend = 000067

3. The modified divisor is subtracted from the modified dividend to determine whether the quotient will fit within the allotted memory space (TB - TA). Quotient overflow will not occur because the result is negative.

Modified dividend	000067
Modified divisor	004230
Result	- 4163

```
BCD division:                  6
                              -1~~7~~
                 04230   000067691
                            4230
                            25391
                            29610
                         99995781
                             4230
                              -11
```

Following command execution:

The contents of the A and B fields are unchanged.

The accumulator contains the result of the division.

Accumulator

Address (hex.)	0140		0141		0142		0143		014D	
Contents	0000	0001	0001	1101	0000	0000	1111	1111	1111	1111
Equivalent	0	1	1	-	0	0	F	F	F	F

014E		014F	
0000	0001	0110	1101
0	1	6	-

The implied B address for a subsequent 4-byte command following the execution of this DIVIDE command is 0140 (00320 dec.).

INOUT (INOUT)

Command code: 112 (70) (F0)

The INOUT command initiates input and output (I/O) operations between the processor and its peripherals; to do this, it uses the peripheral address field (PAF), which is stored in a location indicated by the effective A address, and an S2 status character, which is stored in a location indicated by the effective B address. The T character is not used by this command. The repeat indicator is set OFF by this command if it is on.

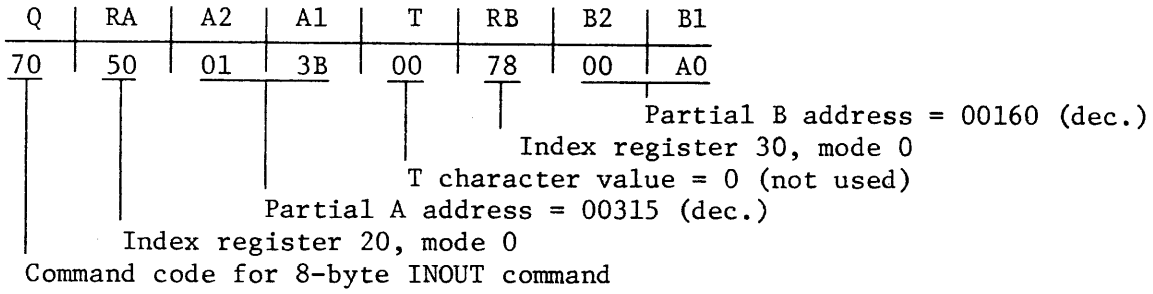
The INOUT command initiates the I/O operation by transmitting the PAF, through the appropriate trunk, to a peripheral which is selected according to the contents of the PAF. The length of the PAF varies depending on the peripheral being selected; however, it must be at least one byte in length. (For further information pertaining to peripheral selection refer in this manual to GENERAL INFORMATION, Processors tab, "NCR Century 101 Processor," under the heading of I/O Functional Operation.)

After the INOUT command transmits the last PAF character, the selected peripheral returns the S2 status character (command initiated) and the INOUT command terminates; the rest of the I/O operation is then handled by the I/O control section of the processor. Certain conditions, however, can cause an S2 status

character (peripheral busy, peripheral on standby, or peripheral inoperative) to be stored before all the PAF has been transmitted; in these instances, the command terminates when the S2 status character is stored. Control is then given to the I/O controller.

The implied B address for a 4-byte command, following the execution of the INOUT command, is the same as the effective B address established prior to command execution.

Example



Assume:

Index register 20 contains 0F00 (03840 dec.)
Index register 30 contains 1002 (04098 dec.)

After command setup:

Effective A address = 013B + 0F00 = 103B (04155 dec.)
Effective B address = 00A0 + 1002 = 10A2 (04258 dec.)

Before command execution:

The A operand address is the address of the first character of a PAF, which will vary in length according to the peripheral desired.

A field address (hex.)	103B	103C	103F
A field contents	0000 0001	0001 000	001 0010

The B operand address points to the memory location of the S2 status character.

B field address (hex.)	10A2
B field contents	0001 0001

During command execution:

1. The PAF (beginning at address 103B) is accessed by the processor, and the selection process begins. If either the trunk or the unit specified by the PAF is busy or if the unit is inoperable, a status character is stored in memory (address 10A2) and the command terminates.

2. If neither the trunk nor the unit is busy and the unit is operable, the processor transmits the remaining portion of the PAF to the peripheral unit, one character at a time.
3. When the peripheral signals that no more PAF characters are required, the processor stores the status character (S2 is stored in address 10A2) and the command terminates; the peripheral continues data transfer under I/O control.
4. If the processor does not receive the required peripheral responses, for any reason, the command terminates and an S2 status character is stored.

Following command execution:

The contents of the A field are unchanged.

The B field contains the S2 status character.

B field address (hex.)	10A2
B field contents	0000 0000

The implied B address for a 4-byte command, following the execution of this INOUT command, is 10A2 (04258 dec.).

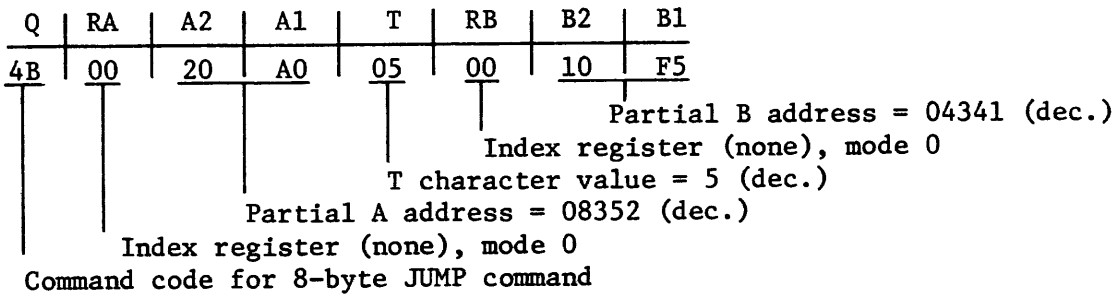
JUMP (JUMP)

Command code: 75 (4B) (CB)

The JUMP command stores the address of the next command in sequence (link address) in the jump link register (IR8), then unconditionally transfers program control to the command indicated by the effective A address. If the effective A address is not 0 mod 4 (evenly divisible by four) a program error occurs, leaving the initial state of the control register unchanged.

The T character and effective B address of the 8-byte JUMP command are not used during command execution; however, they are set up and available as implied operands for a subsequent 4-byte command. The implied B address for a 4-byte command following the execution of the JUMP command is the same as the effective B address established prior to command execution.

Example



After command setup:

Effective A address = 20A0 (08352 dec.)
Effective B address = 10F5 (04341 dec.)

Following command execution:

Program control is transferred to the command at address 20A0 (08352 dec.).

The jump link register (IR8) contains the address of the next command in sequence and the repeat and error indicators are OFF.

The implied B address for a subsequent 4-byte command, following the execution of this JUMP command, is 10F5 (04341 dec.).

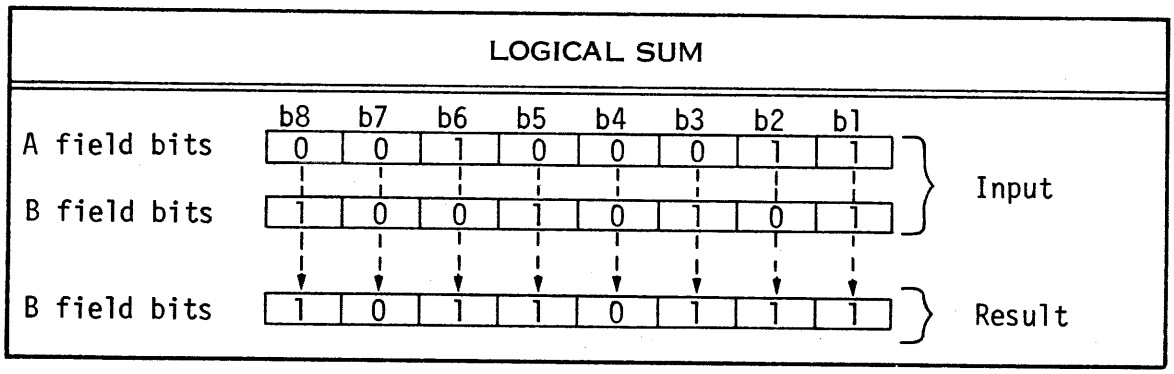
LOGIC (LOGIC)

Command code: 94 (5E) (DE)

The LOGIC command, which is optional on the NCR Century 101 Processor, performs logic operations on the contents of two 1-byte fields (indicated by the effective A and B addresses) as specified by the T character of the command; it then places the result into the B field.

The rightmost four bits (b4 - b1) of the T character specify which of the 16 logic functions is to be performed; the leftmost four bits (b8 - b5) are not used and should be zero to be compatible with other NCR Century processors.

The logic operation, based on Boolean algebra, uses individual bit values of corresponding A and B field bit positions as variables for the equations; the result (as determined by the input and the specified logic function) is stored in the B field bit positions. For example, if the T character specifies a logical sum function and bit 1 in both the A and B fields is ON, bit 1 of the B field is left unchanged and the next two bits (bit 2 of A and bit 2 of B) are compared. If bit 2 of the A field is ON and bit 2 of the B field OFF, the B field bit is set ON. This procedure is repeated for the remaining six bits of each field, with the result replacing the corresponding B field bit.



The implied B address for a subsequent 4-byte command, following the execution of the LOGIC command, is the same as the effective B address established prior to command execution.

The following table contains the 16 logic functions and the final state of the corresponding B field bit (1 or 0) for all possible conditions.

FUNCTIONS OF LOGIC COMMAND

T CODE	FUNCTION	EXPLANATION	A and B INPUT BIT	RESULT	A and B INPUT BIT	RESULT	A and B INPUT BIT	RESULT	A and B INPUT BIT	RESULT
00	B = 0 (CLEAR B)	This logic function sets the B field to binary zeros, regardless of the input from A and B.	A = 0 B = 0	B = 0	A = 1 B = 0	B = 0	A = 0 B = 1	B = 0	A = 1 B = 1	B = 0
01	B = (A+B)' OR B = A'B'	This logic function compares A and B fields by bit; if both bits = 0, the corresponding B bit is set to 1. Any other configuration sets the B bit to 0.	A = 0 B = 0	B = 1	A = 1 B = 0	B = 0	A = 0 B = 1	B = 0	A = 1 B = 1	B = 0
02	B = A'B	This logic function generates the 1's complement of the A field (by bit) and compares each bit to the corresponding B bit. If A is 0 and B is 1, the corresponding B bit is set to 1. Any other configuration sets the corresponding B bit to 0.	A = 0 B = 0	B = 0	A = 1 B = 0	B = 0	A = 0 B = 1	B = 1	A = 1 B = 1	B = 0
03	B = A' (COMPLEMENT A)	This logic function generates the 1's complement of the A field (by bit) and stores the result in the corresponding B field bit.	A = 0 B = 0	B = 1	A = 1 B = 0	B = 0	A = 0 B = 1	B = 1	A = 1 B = 1	B = 0
04	B = AB'	This logic function generates the 1's complement of the B field (by bit) and compares each bit to the corresponding A bit. If A is 1 and B is 0, the corresponding B field bit is set to 1. Any other configuration sets the corresponding B field bit to 0.	A = 0 B = 0	B = 0	A = 1 B = 0	B = 1	A = 0 B = 1	B = 0	A = 1 B = 1	B = 0
05	B = B' (COMPLEMENT B)	This logic function generates the 1's complement of the B field (by bit) and stores the result in the corresponding B field bit.	A = 0 B = 0	B = 1	A = 1 B = 0	B = 1	A = 0 B = 1	B = 0	A = 1 B = 1	B = 0
06	B = AB'+ A'B	This logic function compares the A and B fields by bit; if the bits are unequal in value, the corresponding B bit is set to 1. Any other configuration sets the B bit to 0.	A = 0 B = 0	B = 0	A = 1 B = 0	B = 1	A = 0 B = 1	B = 1	A = 1 B = 1	B = 0
07	B = (AB)' or B = A'+B'	This logic function compares the A and B fields by bit; if both bits = 1, the corresponding B bit is set to 0. Any other configuration sets the B bit to 1.	A = 0 B = 0	B = 1	A = 1 B = 0	B = 1	A = 0 B = 1	B - 1	A = 1 B = 1	B = 0

FUNCTIONS OF LOGIC COMMAND

T CODE	FUNCTION	EXPLANATION	A and B INPUT BIT	RESULT	A and B INPUT BIT	RESULT	A and B INPUT BIT	RESULT	A and B INPUT BIT	RESULT
08	$\underline{B} = AB$	This logic function compares the A and B fields by bit; if both bits = 1, the corresponding B bit is set to 1. Any other configuration sets the B bit to 0.	A = 0 B = 0	B = 0	A = 1 B = 0	B = 0	A = 0 B = 1	B = 0	A = 1 B = 1	B = 1
09	$\underline{B} = A'B' + AB$	This logic function compares the A and B fields by bit; if the bits are equal in value, the corresponding B bit is set to 1. Any other configuration sets the B bit to 0.	A = 0 B = 0	B = 1	A = 1 B = 0	B = 0	A = 0 B = 1	B = 0	A = 1 B = 1	B = 1
0A	$\underline{B} = B$ (STORE B)	This logic function leaves the contents of the B field unchanged.	A = 0 B = 0	B = 0	A = 1 B = 0	B = 0	A = 0 B = 1	B = 1	A = 1 B = 1	B = 1
0B	$\underline{B} = A' + B$	This logic function generates the 1's complement of the A field (by bit) and compares each bit to the corresponding B bit. If A is 1 and B is 0, the corresponding B bit is set to 0. Any other configuration sets the B bit to 1.	A = 0 B = 0	B = 1	A = 1 B = 0	B = 0	A = 0 B = 1	B = 1	A = 1 B = 1	B = 1
0C	$\underline{B} = A$ (STORE A)	This logic function places the contents of the A field into the B field (by bit).	A = 0 B = 0	B = 0	A = 1 B = 0	B = 1	A = 0 B = 1	B = 0	A = 1 B = 1	B = 1
0D	$\underline{B} = A + B'$	This logic function generates the 1's complement of the B field (by bit) and compares each bit to the corresponding A bit. If A is 0 and B is 1, the corresponding B bit is set to 0. Any other configuration sets the B bit to 1.	A = 0 B = 0	B = 1	A = 1 B = 0	B = 1	A = 0 B = 1	B = 0	A = 1 B = 1	B = 1
0E	$\underline{B} = A + B$	This logic function compares A and B fields by bit; if both = 0, the corresponding B bit is set to 0. Any other configuration sets the B bit to 1.	A = 0 B = 0	B = 0	A = 1 B = 0	B = 1	A = 0 B = 1	B = 1	A = 1 B = 1	B = 1
0F	$\underline{B} = 1$ (SET B)	This logic function sets the B field to binary ones, regardless of the inputs from A and B.	A = 0 B = 0	B = 1	A = 1 B = 0	B = 1	A = 0 B = 1	B = 1	A = 1 B = 1	B - 1

Example 1 ("logical product")

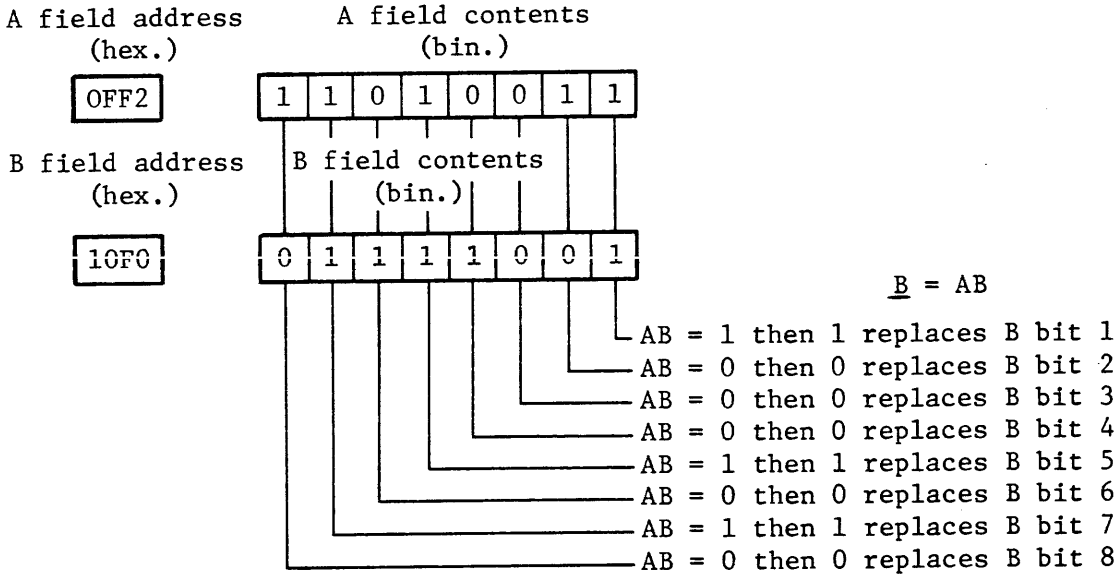
Q	RA	A2	A1	T	RB	B2	B1
5E	00	OF	F2	08	00	10	F0

Partial B address = 04336 (dec.)
Index register (none), mode 0
Function: "logical product" (08)
Partial A address = 04082 (dec.)
Index register (none), mode 0
Command code for 8-byte LOGIC command

After command setup:

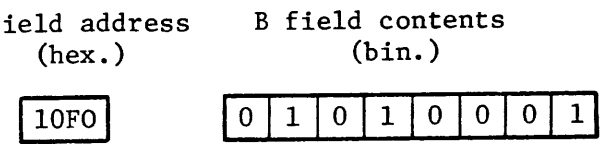
Effective A address = OFF2 (04082 dec.)
Effective B address = 10F0 (04336 dec.)

Before command execution:



Following command execution:

The A field contents are unchanged.



The implied B address for a 4-byte command following the execution of this LOGIC command is 10F0 (04336 dec.).

Example 2 ("exclusive or")

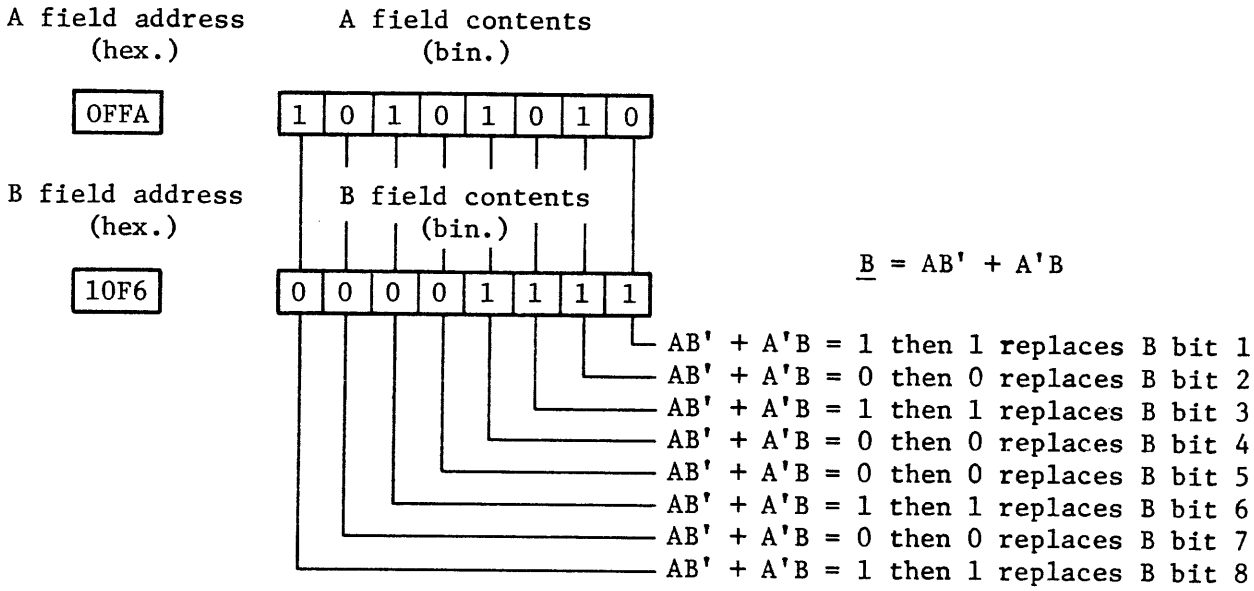
Q	RA	A2	A1	T	RB	B2	B1
5E	00	0F	FA	06	00	10	F6

Partial B address = 04342 (dec.)
Index register (none), mode 0
Function: "exclusive or" (06)
Partial A address = 04090 (dec.)
Index register (none), mode 0
Command code for 8-byte LOGIC command

After command setup:

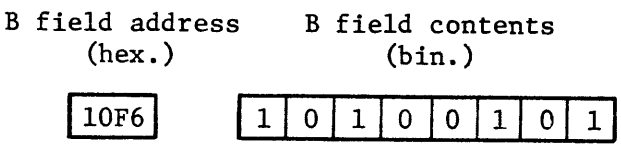
Effective A address = 0FFA (04090 dec.)
Effective B address = 10F6 (04342 dec.)

Before command execution:



Following command execution:

The A field contents are unchanged.



The implied B address for a 4-byte command following the execution of this LOGIC command is 10F6 (04342 dec.).

MOVE A LEFT TO RIGHT (MVAL)

Command code: 84 (54) (D4)

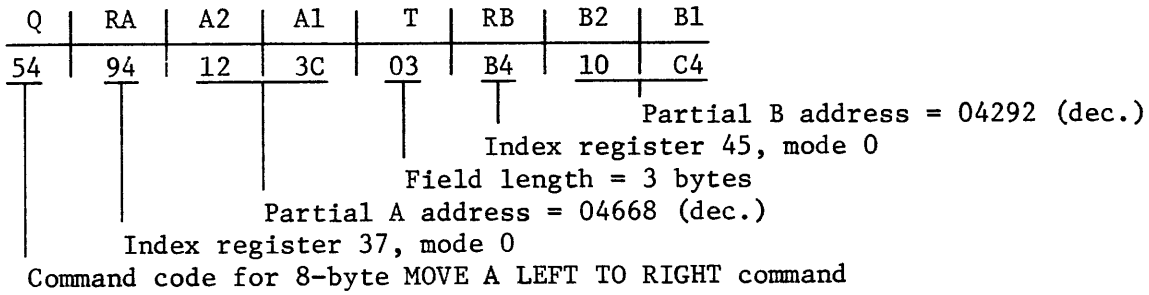
The MOVE A LEFT TO RIGHT command moves the contents of the field specified by the effective A address to the field specified by the effective B address. The lengths of both the A and B fields are indicated by the T character, which may vary in value from 0 through 255, with 0 indicating a length of 256 bytes.

The move takes place from left to right, one byte at a time, the contents of the leftmost byte of the A field replacing the contents of the leftmost byte of the B field; the command terminates when all bytes have been moved.

CAUTION
The results obtained when moving the contents of an A field that overlaps the B field may differ from the results obtained when moving the contents of an A field that does not overlap the B field (assuming that the A and B values are the same in both instances). Using this move command with overlapping fields normally changes the contents of the A field.

The implied B address available for a subsequent 4-byte command following the execution of the MOVE A LEFT TO RIGHT command, is equal to the sum of adding the length of the B field to the effective B address established prior to the execution of this move command.

Example 1



Assume:

Index register 37 contains 00FB (00251 dec.)
Index register 45 contains 01A2 (00418 dec.)

After command setup:

Effective A address = 123C + 00FB = 1337 (04919 dec.)
Effective B address = 10C4 + 01A2 = 1266 (04710 dec.)

Before command execution:

A field address (hex.)	1337	1338	1339
A field contents (ASCII)	0100 0011	0010 1101	0011 0110
Equivalent	C	-	6

B field address (hex.)	1266	1267	1268
B field contents (ASCII)	0100 0000	0100 0111	0100 0110
Equivalent	@	G	F

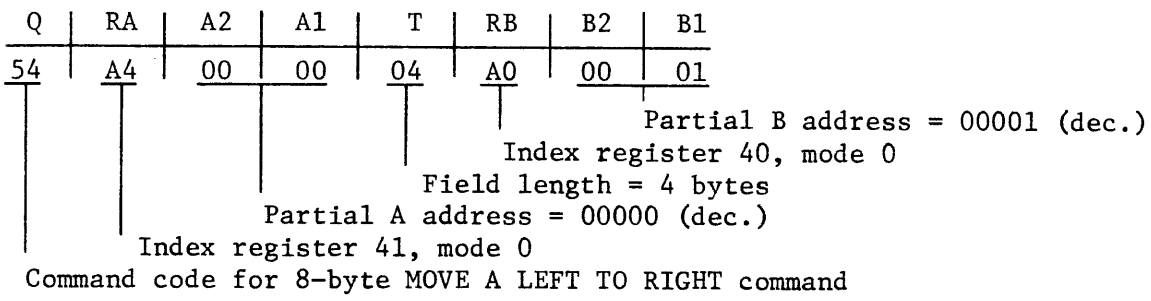
Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	1266	1267	1268
B field contents (ASCII)	0100 0011	0010 1101	0011 0110
Equivalent	C	-	6

The implied B address for a subsequent 4-byte command following the execution of this MOVE A LEFT TO RIGHT command is 1269 (04713 dec.).

Example 2 (Overlapping fields)



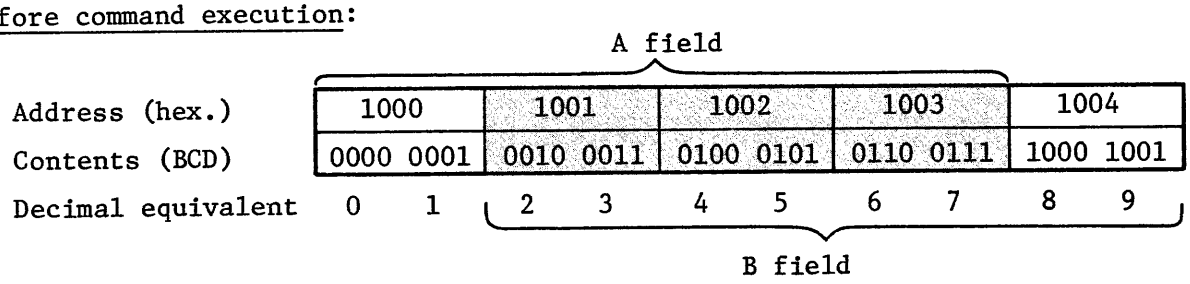
Assume:

Index register 40 contains 1000 (04096 dec.)
Index register 41 contains 1000 (04096 dec.)

After command setup:

Effective A address = 0000 + 1000 = 1000 (04096 dec.)
Effective B address = 0001 + 1000 = 1001 (04097 dec.)

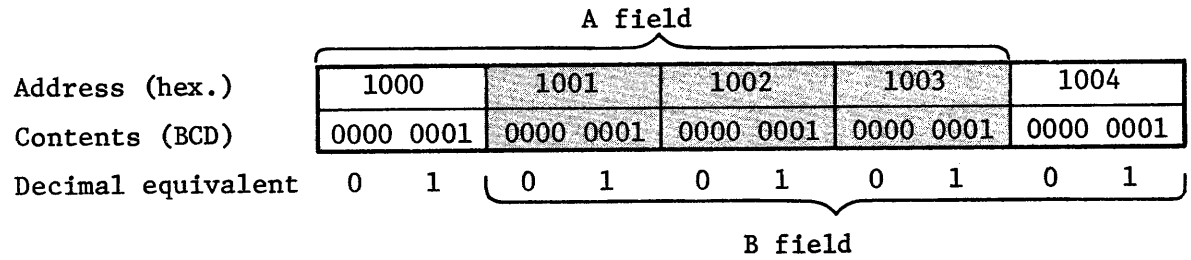
Before command execution:



During execution:

The contents of address 1000 (04096 dec.) replace the contents of address 1001 (04097 dec.) and because of the overlap, the character 0000 0001 will replace the corresponding B field character on each successive move.

Following command execution:



The implied B address for a 4-byte command following the execution of this MOVE A LEFT TO RIGHT command is 1005 (04101 dec.).

MOVE A RIGHT TO LEFT (MVAR)

Command code: 100 (64) (E4)

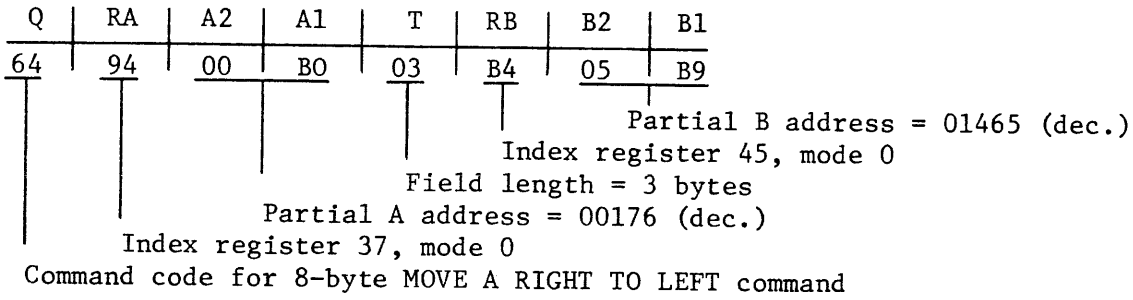
The MOVE A RIGHT TO LEFT command moves the contents of the field specified by the effective A address to the field specified by the effective B address. The lengths of both the A and B fields are indicated by the T character, which may vary in value from 0 through 255, with 0 indicating a length of 256 bytes.

The move takes place from right to left, one byte at a time, the contents of the rightmost A field byte replacing the contents of the rightmost B field byte; the command terminates when all the bytes have been moved.

CAUTION
The results obtained when moving the contents of an A field that overlaps the B field may differ from the results obtained when moving the contents of an A field that does not overlap the B field (assuming that the A and B field values are the same in both instances). Using this move command with overlapping fields normally changes the contents of the A field.

The implied B address available for a subsequent 4-byte command, following the execution of the MOVE A RIGHT TO LEFT command, is the same as the effective B address established prior to command execution.

Example 1



Assume:

Index register 37 contains 0E74 (03700 dec.)
Index register 45 contains 1194 (04500 dec.)

After command setup:

Effective A address = 00B0 + 0E74 = 0F24 (03876 dec.)
Effective B address = 05B9 + 1194 = 174D (05965 dec.)

Before command execution:

A field address (hex.)	0F24	0F25	0F26
A field contents (ASCII)	0100 0011	0010 1101	0011 0110
Equivalent	C	-	6

B field address (hex.)	174D	174E	174F
B field contents (ASCII)	0011 0001	0011 0010	0011 0011
Equivalent	1	2	3

Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	174D	174E	174F
B field contents (ASCII)	0100 0011	0010 1101	0011 0110
Equivalent	C	-	6

The implied B address for a 4-byte command following the execution of this MOVE A RIGHT TO LEFT command is 174D (05965 dec.).

Example 2 (Fields overlap)

Q	RA	A2	A1	T	RB	B2	B1
64	A0	00	C1	04	A0	00	00

Partial B address = 00000
Index register 40, mode 0
Field length = 4 bytes
Partial A address = 00001 (dec.)
Index register 40, mode 0
Command code for 8-byte MOVE A RIGHT TO LEFT command

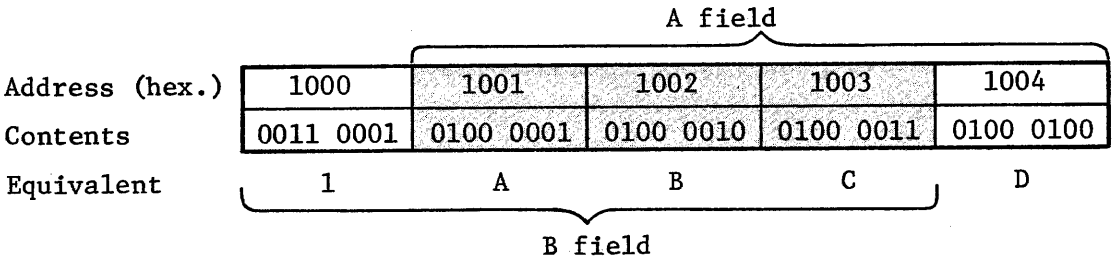
Assume:

Index register 40 contains 1000 (04096 dec.)

After command setup:

Effective A address = 0001 + 1000 = 1001 (04097 dec.)
Effective B address = 0000 + 1000 = 1000 (04096 dec.)

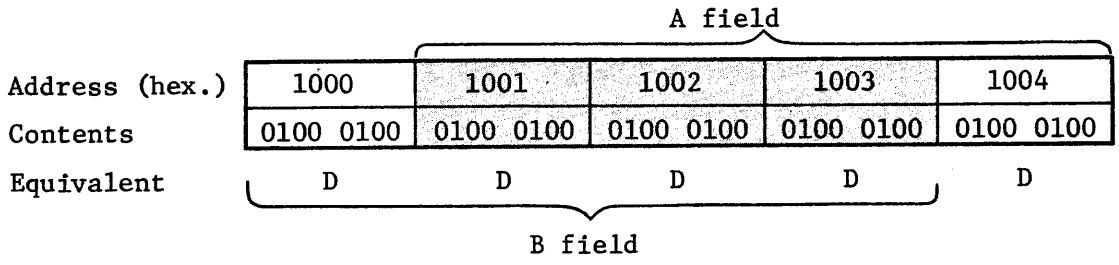
Before command execution:



During execution:

The contents of address 1004 (04100 dec.) replace the contents of address 1003 (04099 dec.) and because of the overlap, the character 0100 0100 will replace the corresponding B field character on each successive move.

Following command execution:



The implied B address for a 4-byte command following the execution of this MOVE TO RIGHT TO LEFT command is 1000 (04096 dec.).

MOVE B RIGHT TO LEFT (MVBR)

Command code: 68 (44) (C4)

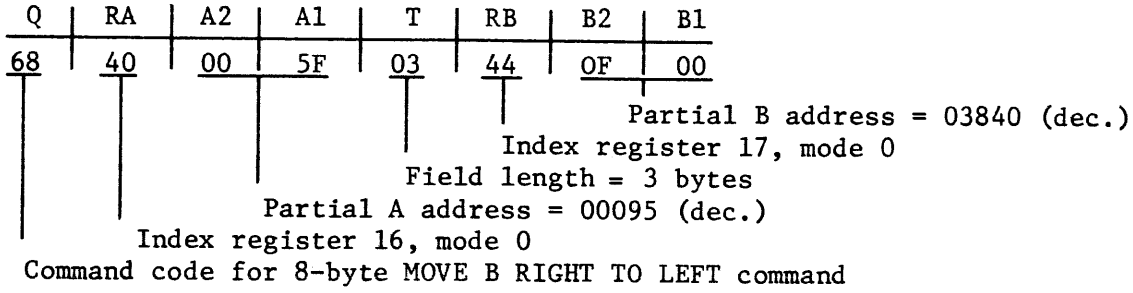
The MOVE B RIGHT TO LEFT command moves the contents of the field specified by the effective B address to the field specified by the effective A address. The lengths of both the A and B fields are indicated by the T character, which may vary in value from 0 through 255, with 0 indicating a length of 256 bytes.

The move takes place from right to left, one byte at a time, the contents of the rightmost B field byte replacing the contents of the rightmost byte of the A field; the command terminates when all bytes have been moved.

CAUTION
The results obtained when moving the contents of a B field that overlaps the A field may differ from the results obtained when moving the contents of a B field that does not overlap the A field (assuming that the A and B values are the same in both instances). Using this move command with overlapping fields normally changes the contents of the B field.

The implied B address for a subsequent 4-byte command, following the execution of the MOVE B RIGHT TO LEFT command, is equal to the effective B address established prior to command execution.

Example 1



Assume:

Index register 16 contains 1201 (04609 dec.)
Index register 17 contains 0415 (01045 dec.)

After command setup:

Effective A address = 005F + 1201 = 1260 (04704 dec.)
Effective B address = 0F00 + 0415 = 1315 (04885 dec.)

Before command execution:

A field address (hex.)	1260	1261	1262
A field contents (ASCII)	0100 0110	0100 1001	0100 0101
Equivalent	F	I	E

B field address (hex.)	1315	1316	1317
B field contents (ASCII)	0100 0010	0100 0011	0100 0100
Equivalent	B	C	D

Following command execution:

A field address (hex.)	1260	1261	1262
A field contents (ASCII)	0100 0010	0100 0011	0100 0100
Equivalent	B	C	D

The contents of the B field are unchanged.

The implied B address for a subsequent 4-byte command, following the execution of this MOVE B RIGHT TO LEFT command, is 1315 (04885 dec.).

Example 2 (Fields overlap)

Q	RA	A2	A1	T	RB	B2	B1
68	40	00	00	04	40	00	01

Partial B address = 00001 (dec.)
Index register 16, mode 0
Field length = 4 bytes
Partial A address = 00000
Index register 16, mode 0
Command code for 8-byte MOVE B RIGHT TO LEFT command

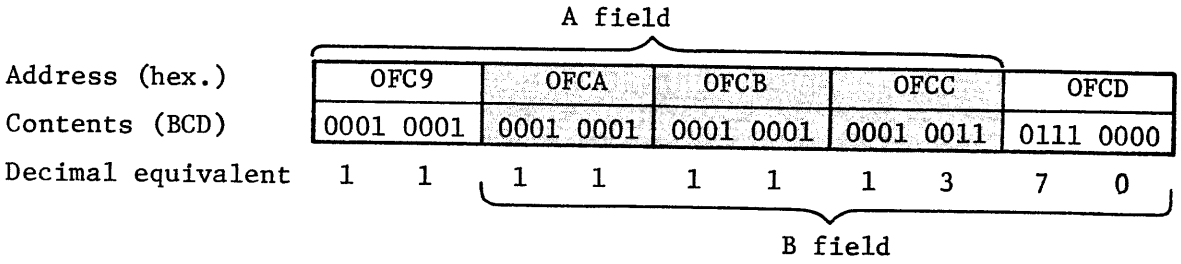
Assume:

Index register 16 contains OFC9 (04041 dec.)

After command setup:

Effective A address = 0000 + OFC9 = OFC9 (04041 dec.)
Effective B address = 0001 + OFC9 = OFCA (04042 dec.)

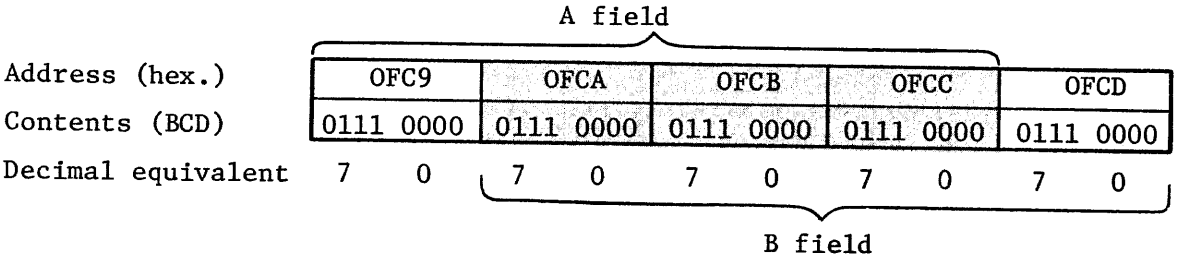
Before command execution:



During execution:

The contents of address OFCD (04045 dec.) replace the contents of address OFCC (04044 dec.). Because of the overlap, the character 0111 0000 replaces the corresponding A field character on each successive move.

Following command execution:



The implied B address for a subsequent 4-byte command, following the execution of this MOVE B RIGHT TO LEFT command, is OFCA (04042 dec.).

MULTIPLY (PMULT)

Command code: 93 (5D) (DD)

The MULTIPLY command, which is optional with the NCR Century 101 Processor, multiplies the contents of the field specified by the effective B address (multiplicand) by the contents of the field specified by the effective A address (multiplier). The result (product) is stored in the memory accumulator. The contents of the A and B fields are assumed to be signed packed data (see PACK command description) with the signs stored in the rightmost four bits of each field. If any position except the sign position has a value other than 0 through 9, a program error occurs.

The leftmost four bits of the T character (TA), which indicate the length of the A field, may vary in value from 0 through 7, with 0 indicating 8 bytes. A program error occurs if the leftmost bit (b8) of TA is ON. The rightmost four bits of the T character (TB), which indicate the length of the B field, may vary in value from 0 through 7, with 0 indicating 8 bytes. A program error occurs if the leftmost bit (b4) of TB is ON.

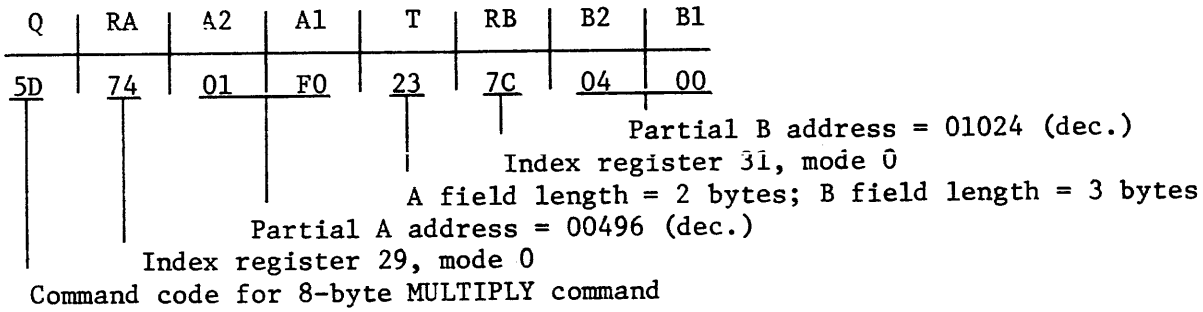
The result (product) of the multiplication is stored as signed packed data in the memory accumulator, a fixed area in memory located at decimal addresses 320 through 335 (hex. addresses 140-14F). The sign of the product is determined by algebraic laws; that is, a positive sign (bit configuration 1011) is stored when the A and B fields have like signs, and a negative sign (bit configuration 1101) is stored when the fields have unlike signs. The product is stored (right-justified and zero-filled to the left) beginning at the memory location established by subtracting the length of the product field (TA + TB) from 336, and terminates at memory location 335 (decimal).

CAUTION
If either the A or B field occupies any part of the memory accumulator an indeterminate product results.

Multiplication is achieved by multiplying the entire B field by each A field digit (excluding the signs) and adding the resulting partial products together to arrive at the final product. For example, the result of multiplying the B field by the rightmost A field digit (using logic circuitry) is stored in the memory accumulator; the next partial product (established by using the next multiplier digit to the left), is shifted to the left one digit position and added to the contents of the memory accumulator. This process is repeated until the entire product is established and located in the memory accumulator.

The implied B address available for a subsequent 4-byte command, following the execution of the MULTIPLY command, is equal to the result of subtracting the length of the A field from 335 (dec.). The implied T character is equal to the binary value of the T character established prior to command execution.

Example



Assume:

Index register 29 contains 0B54 (02900 dec.)
Index register 31 contains 0C1C (03100 dec.)

After command setup:

Effective A address = 01F0 + 0B54 = 0D44 (03396 dec.)
Effective B address = 0400 + 0C1C = 101C (04124 dec.)

Before command execution:

A field address (hex.)	OD44		OD45	
A field contents	0000	0001	0101	1011
Decimal equivalent	0	1	5	+

B field address (hex.)	101C		101D		101E	
B field contents	0000	0001	1000	0101	0000	1011
Decimal equivalent	0	1	8	5	0	+

Arithmetic manipulation:

Decimal equivalent of multiplication to be performed:

```
1850+
 x 15+
 9250
1850
27750+
```

Processor multiply logic:

1. Multiplicand (B field contents) 01850+
   Multiplier (A field contents) x 015+

2. The partial product established by multiplying the rightmost multiplier digit (5 x 1850) is stored in the memory accumulator. 9250

3. The partial product established by multiplying the B field by the next multiplier digit to the left (1 x 1850), is shifted to the left one digit position and added to the contents of the memory accumulator. 1850

4. The final product is located in the memory accumulator. 27750+

Following command execution:

The contents of the A and B fields are unchanged.

Accumulator address (hex.)	014B		014C		014D	
Accumulator contents	0000	0000	0000	0000	0010	0111
Decimal equivalent	0	0	0	0	2	7

	014E		014F	
	0111	0101	0000	1011
	7	5	0	+

The implied B address for a subsequent 4-byte, command following the execution of this MULTIPLY command, is 014D (00333 dec.).

OPTION SWITCHES INPUT (SWIN)

Command code: 80 (50) (DO)

The OPTION SWITCHES INPUT command should not be used in applications programming; it is intended for software use only. The NCR Century 101 Processor sets the OPTION SWITCHES INPUT command up (establishes the effective A and B addresses, and T character), but does not execute it; program control is then transferred to the next command in sequence.

PACK (PACK)

Command code: 76 (4C) (CC)

The PACK command compacts the contents of the field specified by the effective A address (by ignoring bits 8 through 5) and stores the result in the field specified by the effective B address. The A field may contain any combination of the characters shown on the following chart; however, only the characters indicated by the shaded area can be restored to their original values by the UNPACK command (see UNPACK command description).

NCR CENTURY CODE CHART

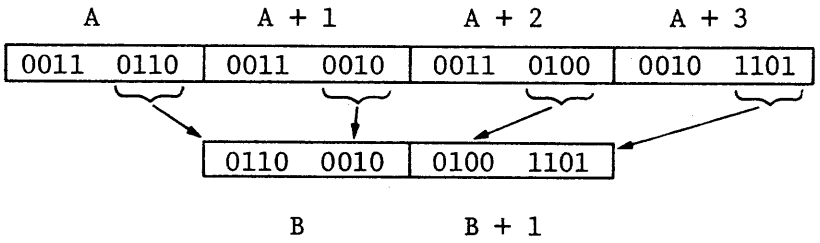
B_8-B_5 ↓ / B_4-B_1 →		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	NL/LF	VT	FF	CR	SO	SI
0001	1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010	2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0011	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0110	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	7	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	DEL

The length of the A field is indicated by the T character, which may vary in value from 0 through 255, with 0 indicating a length of 256 bytes. The length of the B field depends upon the state of the T character and is determined in the following manner:

1. If the T character is zero, the B field is 128 bytes in length.
2. If the value of the T character is an even number, the length of the B field is equal to one-half the length of the A field.
3. If the value of the T character is an odd number, the length of the B field is equal to one-half of the result from adding one to the length of the A field.

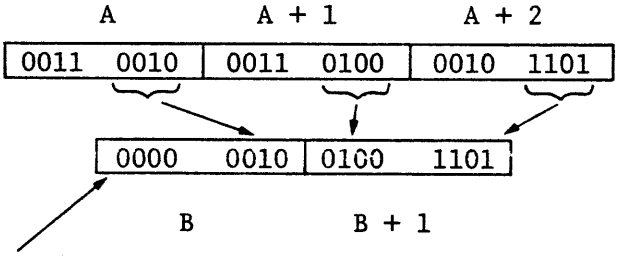
Packing is performed sequentially from left to right, beginning with the leftmost A and B field memory locations; however, the actual B field replacement location (within a byte) varies, depending on the state of the T character value. If the A field is an even number of bytes in length (T even), the rightmost four bits of the first A field byte are placed in the leftmost four bits of the first B field byte, and the rightmost four bits of the next A field byte are placed in the rightmost four bits of the first B field byte. This process is repeated until all of the A field characters are packed into the B field.

T even:



If the A field is an odd number of bytes in length (T odd), the leftmost four bits of the first B field byte are set to zero, and the rightmost four bits of the first A field byte are placed into the rightmost four bits of the first B field byte. The rightmost four bits of the second A field byte are placed into the leftmost four bits of the second B field byte, and the rightmost four bits of the third A field byte are placed into the rightmost four bits of the second B field byte. This process is repeated until all of the A field characters are packed into the B field.

T odd:

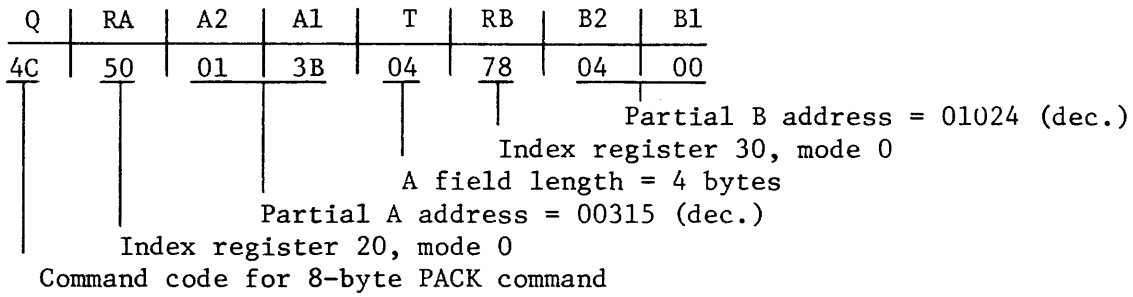


When the command is completed, the B field contains packed data and the contents of the A field are unchanged.

CAUTION
The result obtained when using the PACK command with overlapping A and B fields may differ from the result obtained when working with fields that do not overlap (assuming the same A and B field values are used in both instances); the contents of the A field are normally changed when using overlapping fields.

The implied B address for a subsequent 4-byte command is equal to the sum of the length of the B field and the effective B address that was established prior to the execution of the PACK command.

Example 1 (T is even)



Assume:

Index register 20 contains 0F02 (03842 dec.)
Index register 30 contains 0BB8 (03000 dec.)

After command setup:

Effective A address = 013B + 0F02 = 103D (04157 dec.)
Effective B address = 0400 + 0BB8 = 0FB8 (04024 dec.)

Before command execution:

A field address (hex.)	103D	103E	103F	1040
A field contents (ASCII)	0011 0000	0011 0011	0011 0010	0011 0100
Equivalent	0	3	2	4

The contents of the B field are not significant.

Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	0FB8		0FB9	
B field contents (packed)	0000	0011	0010	0100
Equivalent	0	3	2	4

The implied B address for a 4-byte command following the execution of this PACK command is 0FBA (03002 dec.).

Example 2 (T is odd)

Q	RA	A2	A1	T	RB	B2	B1
4C	64	01	F4	03	40	00	C8

Partial B address = 00200 (dec.)
Index register 16, mode 0
A field length = 3 bytes
Partial A address = 00500 (dec.)
Index register 25, mode 0
Command code for 8-byte PACK command

Assume:

Index register 16 contains 0EF8 (03832 dec.)
Index register 25 contains 0F1A (03866 dec.)

After command setup:

Effective A address = 01F4 + 0F1A = 110E (04366 dec.)
Effective B address = 00C8 + 0EF8 = 0FC0 (04032 dec.)

Before command execution:

	110E		110F		1110	
A field address (hex.)						
A field contents (ASCII)	0011	0100	0011	0011	0011	0010
Equivalent	4		3		2	

The contents of the B field are not significant.

Following command execution:

The contents of the A field are unchanged.

	0FC0		0FC1	
B field address (hex.)				
B field contents (packed)	0000	0100	0011	0010
Equivalent	0	4	3	2

The implied B address for a 4-byte command following the execution of this PACK command is 0FC2 (04034 dec.).

REPEAT (REPEAT)

Command code: 102 (66) (E6)

The REPEAT command sets up conditions that cause the command immediately following the REPEAT command to be executed the number of times (repeat number) indicated by the contents of the 1-byte field specified by the effective A address. The repeat number may vary in value from 0 through 255, with 0 indicating that the command following is to be skipped. The T character and B operand are not used by the REPEAT command; however, the T character and effective B address are set up and available to a subsequent 4-byte command as an implied B address and implied T character.

In effect, the REPEAT command moves the contents of the field specified by the effective A address (repeat number) to the repeat counter at memory location 0020 (hex.). If the repeat number is greater than zero, the repeat indicator is set ON and program control is transferred to the command that is to be repeated. When the repeat number is equal to zero, the initial state of the repeat indicator remains unchanged and program control is transferred to the second command following the REPEAT command.

Any command may follow a REPEAT command; however, some commands either conditionally or unconditionally nullify the repeat operation by setting the repeat indicator OFF. The commands that unconditionally nullify the repeat operation follows:

1. BRANCH (any branch command)
2. INOUT
3. JUMP
4. SET IP OFF
5. SET IP ON
6. WAIT

The commands that conditionally nullify the repeat operation are:

1. COMPARE BINARY, when the contents of the A field are equal to or greater in value than the contents of the B field.

2. COMPARE SIGNED, when the contents of the A field are equal to or greater in value than the contents of the B field.

3. TEST BIT, when either the T character is zero or when the one bits in the T character are matched by corresponding one bits in the B field.

4. TEST CHARACTER EQUAL, when the characters compared are identical.

5. TEST CHARACTER UNEQUAL, when the characters compared are not identical.

The REPEAT command and the subsequent repeating operation are as follows:

1. The repeat number is stored in the repeat counter and the repeat indicator is set ON.

2. The repeat number is compared to zero. If they are equal, the repeat indicator is set OFF and program control is transferred to the second command following the REPEAT command. If they are not equal, step 3 is initiated.

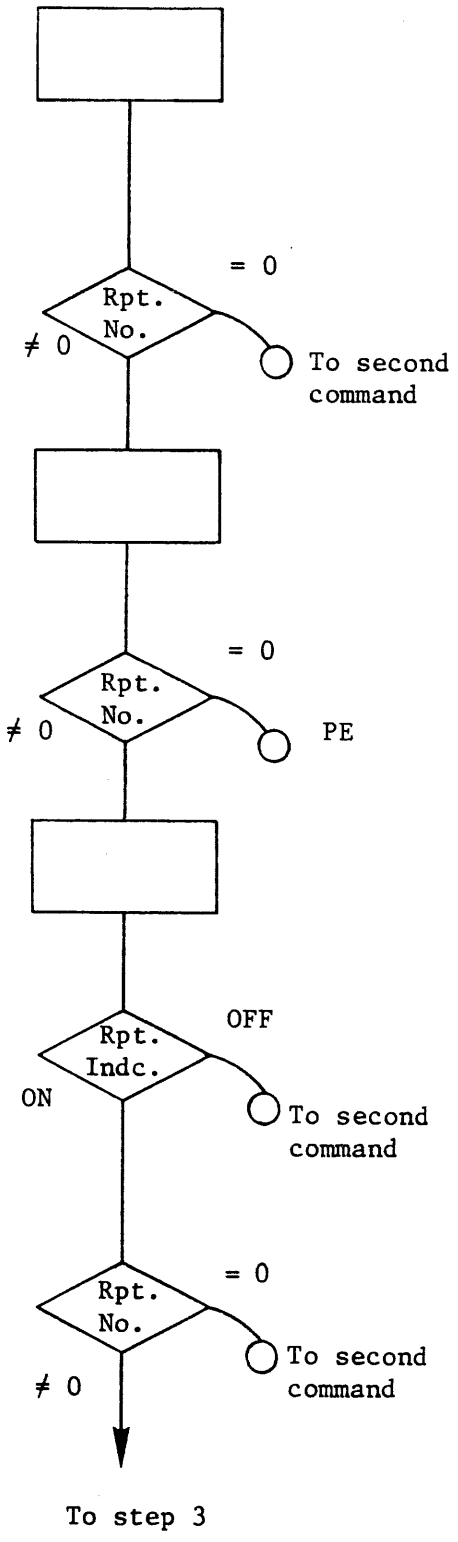
3. The command to be repeated is set up and executed.

4. The repeat number is compared to zero. If they are equal, a program error occurs. If they are not equal, step 5 is initiated.

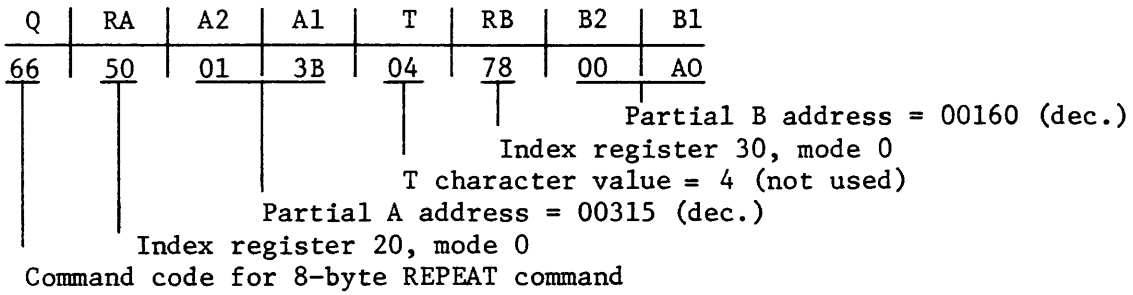
5. The repeat number is decremented by one.

6. The repeat indicator is tested. If it is OFF (due to either a conditional or unconditional nullifying command), program control is transferred to the second command following the REPEAT command. If the repeat indicator is ON, step 7 is initiated.

7. The repeat number is compared to zero. If they are equal, the repeat indicator is set OFF and program control transferred to the second command following the REPEAT command. If they are not equal, step 3 is initiated.



Example



Assume:

Index register 20 contains OFFE (04094 dec.)
Index register 30 contains 10E0 (04320 dec.)

After command setup:

Effective A address = 013B + OFFE = 1139 (04409 dec.)
Effective B address = 00A0 + 10E0 = 1180 (04480 dec.)

Before command execution:

A field address (hex.)	1139
A field contents	00000110
Decimal equivalent	6

The B field is not used.

Following command execution:

The repeat indicator is set ON, and the contents of the A field are placed in the repeat counter.

The contents of the A field are unchanged.

The implied B address available for a subsequent 4-byte command, following the execution of this REPEAT command is 1180 (04480 dec.).

RESTORE (RESTOR)

Command code: 72 (48) (C8)

The RESTORE command allows the status of the processor (the contents of various hardware registers and the states of various flags and indicators) to be changed according to the contents of an 8-byte field, called the status word. The relationship between the status word and the processor values affected are as follows:

STATUS WORD

BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6	BYTE 7	BYTE 8
T CHARACTER	NOT USED	EFFECTIVE B ADDRESS		FLAGS & INDICATORS	NOT USED	ADDRESS FOR CONTROL REGISTER *	

BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1
NOT USED	NOT USED	OVERFLOW FLAG	REPEAT INDICATOR	NOT USED	GREATER FLAG	EQUAL FLAG	LESS FLAG

* The address contained in bytes 7 and 8 must be 0 mod 4 (evenly divisible by 4) or a program error occurs leaving the initial contents of the control register unchanged.

The beginning address of the status word is indicated by the effective A address of the RESTORE command and must be 0 mod 4 or a program error occurs. The B operand and the T character portions of the 8-byte command format are not used by the RESTORE command.

Following the execution of the RESTORE command, the processor flags and indicators are set according to the contents of byte 5 of the status word; program control is then transferred to the command indicated by the address contained in bytes 7 and 8. The implied T character and implied B address available to a subsequent 4-byte command are identical to the values contained in bytes 1, 3, and 4 of the status word.

Example

Q	RA	A2	A1
C8	00	14	00

Effective A address = 05120 (dec.)
Index register (none), mode 0
Command code for 4-byte RESTORE command

After command setup:

Effective A address = 1400 (05120 dec.)

Before command execution:

A field address (hex.)

A field contents

Equivalent

1400	1401	1402	1403
0000 0110	0000 0000	0001 0000	1111 0101
0 6	0 0	1 0	F 5

1404	1405	1406	1407
0000 0100	0000 0000	0001 0001	0110 1100
0 4	0 0	1 1	6 C

Following command execution:

The contents of the A field are unchanged, the greater flag is ON, and the control register contains 116C (04460 dec.).

The implied B address and T character available to a subsequent 4-byte command are 10F5 (04341 dec.) and 06, respectively.

SCAN ON KEY EQUAL (SCANE)

Command Code: 86 (56) (D6)

The field specified by the effective A address is a 1-character field called the scan key. The field specified by the effective B address is the field to be scanned. The T character specifies the number of bytes in the B field. T may be any number from 0 through 255, with 0 being equivalent to 256.

The SCAN ON KEY EQUAL command compares the scan key binarily to each character of the B field, starting with the leftmost character, until it finds a B field character that is equal to the scan key, or until the B field is exhausted.

If the scan key is found to be equal to a B field character, the address, plus one, of that character is stored in index register nine. The E flag is turned ON, and the repeat indicator (RI) is turned OFF following the command.

If the scan key is not equal to any B field character, the address B+T is stored in IR9, except when T=0; in that case B+256 is stored in IR9. The G flag or the L flag is turned ON, depending on whether the scan key is greater or less than the rightmost B field character. The repeat indicator is left undisturbed.

The implied B address for a subsequent 4-byte command varies depending on the cause of command termination. If the scan key is found to be equal to a B field character, the implied B address is pointing one byte beyond that character. If no B field character satisfies the scan condition, the implied B address is equal to the sum of the B field length and the effective B address. In either case, the address is stored in both the appropriate hardware register and in index register nine.

Example

Before command execution, assume the values to be:

Effective A address = 1800
Effective B address = 1820
T (length of B field) = 5

A field address (hex.)	1800
A field contents (bin.)	00110111
Decimal equivalent	7

B field address (hex.)	1820	1821	1822	1823	1824
B field contents (bin.)	00110111	00111000	00111001	00110010	00110011
Decimal equivalent	7	8	9	2	3

Following command execution:

Since the scan key (7) in the A field is equal to the content of address 1820 (7) in the B field, the "equal to" condition is satisfied. The E flag is turned ON, the address, plus one, of the B field character satisfying the scan condition (1821) is stored in IR9, and the repeat indicator (RI) is turned OFF following the command.

The implied B address for a subsequent 4-byte command, following the execution of this SCAN ON KEY EQUAL COMMAND, is 1821.

SCAN ON KEY GREATER THAN (SCANG)

Command Code: 87 (57) (D7)

The field specified by the effective A address is a 1-character field called the scan key. The field specified by the effective B address is the field to be scanned. The T character specifies the number of bytes in the B field. T may be any number from 0 through 255, with 0 being equivalent to 256.

The SCAN ON KEY GREATER THAN command compares the scan key binarily to each character of the B field, starting with the leftmost character, until it finds a B field character that is less than the scan key (the scan key is in a "greater than" condition), or until the B field is exhausted.

If the scan key is greater than a B field character, the address, plus one, of that character is stored in index register nine. The G flag is turned ON, and the repeat indicator (RI) is turned OFF following the command.

If the scan key is not greater than any B field character, the address B+T is stored in IR9, except when T=0; in that case B+256 is stored in IR9. The E flag or the L flag is turned ON, depending on whether the scan key is equal to or less than the rightmost B field character. The repeat indicator is left undisturbed.

The implied B address for a subsequent 4-byte command varies depending on the cause of command termination. If the scan key is found to be greater than a B field character, the implied B address is pointing one byte beyond that character. If no B field character satisfies the scan condition, the implied B address is equal to the sum of the B field length and the effective B address. In either case, the address is stored in both the appropriate hardware register and in index register nine.

Example

Before command execution, assume the values to be:

Effective A address = 1800
Effective B address = 1820
T (length of B field) = 5

A field address (hex.)	1800
A field contents (bin.)	00110111
Decimal equivalent	7

B field address (hex.)	1820	1821	1822	1823	1824
B field contents (bin.)	00110111	00111000	00111001	00110010	00110011
Decimal equivalent	7	8	9	2	3

Following command execution:

Since the scan key (7) in the A field is greater than the content of address 1823 (2) in the B field, the "greater than" condition is satisfied. The G flag is turned ON, the address, plus one, of the B field character satisfying the scan condition (1824) is stored in IR9, and the repeat indicator (RI) is turned OFF following the command.

The implied B address for a subsequent 4-byte command, following the execution of this SCAN ON KEY GREATER THAN command, is 1824.

SCAN ON KEY LESS THAN (SCANL)

Command Code: 85 (55) (D5)

The field specified by the effective A address is a 1-character field called the scan key. The field specified by the effective B address is the field to be scanned. The T character specifies the number of bytes in the B field. T may be any number from 1 through 255, with 0 being equivalent to 256.

The SCAN ON KEY LESS THAN command compares the scan key binarily to each character of the B field, starting with the leftmost character, until it finds a B field character that is greater than the scan key (the scan key is in a "less than" condition), or until the B field is exhausted.

If the scan key is less than a B field character, the address, plus one, of that character is stored in index register nine. The L flag is turned ON, and the repeat indicator (RI) is turned OFF following the command.

If the scan key is not less than any B field character, the address B+T is stored in IR9, except when T=0; in that case B+256 is stored in IR9. The E flag or the G flag is turned ON, depending on whether the scan key is equal to or greater than the rightmost B field character. The repeat indicator is left undisturbed.

The implied B address for a subsequent 4-byte command varies depending on the cause of command termination. If the scan key is found to be less than a B field character, the implied B address is pointing one byte beyond that character. If no B field character satisfies the scan condition, the implied B address is equal to the sum of the B field length and the effective B address. In either case, the address is stored in both the appropriate hardware register and in index register nine.

Example

Before command execution, assume the values to be:

Effective A address = 1800
Effective B address = 1820
T (length of B field) = 5

A field address (hex.)	1800
A field contents (bin.)	00110111
Decimal equivalent	7

B field address (hex.)	1820	1821	1822	1823	1824
B field contents (bin.)	00110111	00111000	00111001	00110010	00110011
Decimal equivalent	7	8	9	2	3

Following command execution:

Since the scan key (7) in the A field is less than the content of address 1821 (8) in the B field, the "less than" condition is satisfied. The L flag is turned ON, the address, plus one, of the B field character satisfying the scan condition (1822) is stored in IR9, and the repeat indicator (RI) is turned OFF following the command.

The implied B address for a subsequent 4-byte command, following the execution of this SCAN ON KEY LESS THAN command, is 1822.

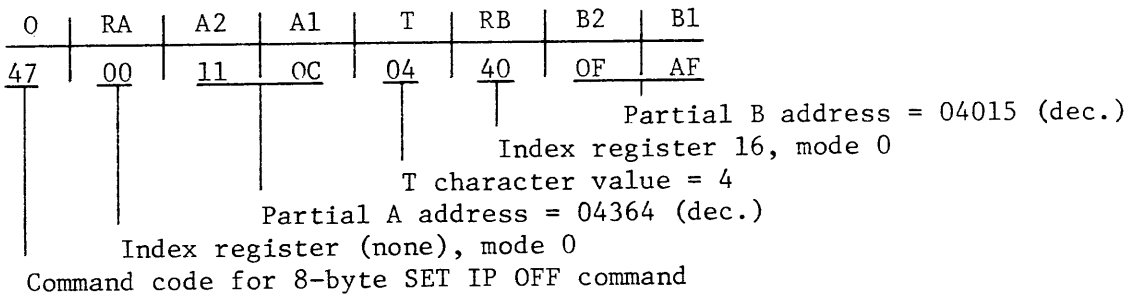
SET IP OFF (IPOFF)

Command code: 71 (47) (C7)

The SET IP OFF command sets the interrupt permit and repeat indicators OFF and transfers program control to the command specified by its effective A address. The effective A address must be 0 mod 4 (evenly divisible by four) or a program error occurs, leaving the control register undisturbed.

The T character and effective B address are not used by this command; however, they are set up and available as implied operands for a subsequent 4-byte command. The implied B address for a subsequent 4-byte command is the same as the effective B address established prior to command execution.

Example



Assume:

Index register 16 contains 00F0 (00240 dec.)

After command setup:

Effective A address = 110C (04364 dec.)
Effective B address = 0FAF + 00F0 = 109F (04255 dec.)

Following command execution:

The interrupt permit and repeat indicators are set OFF, and the implied B address for a subsequent 4-byte command is 109F (04255 dec.).

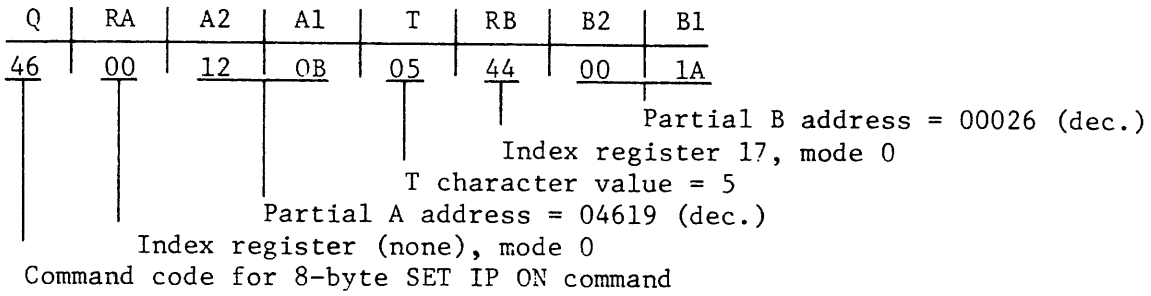
SET IP ON (IPON)

Command code: 70 (46) (C6)

The SET IP ON command, sets the interrupt permit indicator ON, sets the repeat indicator OFF, and transfers program control to the command specified by the effective A address. The effective A address must be 0 mod 4 (evenly divisible by four) or a program error occurs. On termination of the SET IP ON command, the test for interrupt trapping is bypassed.

The T character and effective B address are not used by this command; however, they are set up and available as implied operands. The implied B address for a subsequent 4-byte command is the same as the effective B address established prior to the execution of the SET IP ON command.

Example



Assume:

Index register 17 contains 100A (04106 dec.)

After command setup:

Effective A address = 120B (04619 dec.)
Effective B address = 001A + 100A = 1024 (04132 dec.)

Following command execution:

The contents of the A field are unchanged, the interrupt permit indicator is ON, the repeat indicator is OFF, and program control is transferred to the command specified by the effective A address.

The implied B address for a subsequent 4-byte command, following the execution of this SET IP ON command, is 1024 (04132 dec.).

SUBTRACT BINARY (BSUB)

Command code: 97 (61) (E1)

The SUBTRACT BINARY command, binarily subtracts the contents of the field specified by the effective A address from the contents of the field specified by the effective B address and stores the result in the B field. The A and B fields are assumed to contain unsigned, positive binary data. The lengths of both the A and B fields are indicated by the T character, which may vary in value from 0 through 255, with 0 indicating a length of 256 bytes.

Subtraction is performed (using complementary addition) from right to left, one byte at a time, with a carry from one byte being added to the byte on its left. A carry beyond the leftmost byte does not effect the state of the overflow flag and is not included in the result of the subtraction. If the A field value is greater than the B field value, the result is stored as the 2's complement.

The result obtained when subtracting the contents of overlapping A and B fields may differ, depending upon the contents, from the result obtained when subtracting the contents of A and B fields that do not overlap (assuming that the same A and B field values are used in both instances). The contents of the A field are normally changed when using overlapping fields.

The implied B address for a subsequent 4-byte command, following the execution of the SUBTRACT BINARY command, is the same as the effective B address established prior to the execution of the SUBTRACT BINARY command.

Example 1 (A field value less than B field value)

Q	RA	A2	A1	T	RB	B2	B1
61	00	10	0A	02	00	10	FB

Partial B address = 04347 (dec.)
Index register (none), mode 0
Field length = 2 bytes
Partial A address = 04106 (dec.)
Index register (none), mode 0
Command code for 8-byte SUBTRACT BINARY command

After command setup:

Effective A address = 100A (04106 dec.)
Effective B address = 10FB (04347 dec.)

Before command execution:

A field address (hex.)	100A	100B
A field contents (bin.)	00000000	00101000

B field address (hex.)	10FB	10FC
B field contents (bin.)	00010010	01101100

Arithmetic manipulation:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost bit of the A field 1's complement.

1's complement	11111111	11010111
Initial carry		1
	11111111	11010110
Carries		111
2's complement	11111111	11011000

2. The B field contents are added to the 2's complement of the A field.

2's complement	11111111	11011000
B field	00010010	01101100
	11101101	10110100
Carries	11111111	1111
Final result	1 00010010	01000100

The carry beyond the specified field length is ignored.

Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	10FB	10FC
B field contents (bin.)	00010010	01000100

The implied B address for a subsequent 4-byte command, following the execution of this SUBTRACT BINARY command, is 10FB (04347 dec.).

Example 2 (A field value greater than B field value)

Assume that the fields used in example 1 contain the following information before command execution:

A field address (hex.)	100A	100B
A field contents (bin.)	01110101	10100100

B field address (hex.)	10FB	10FC
B field contents (bin.)	01000011	01101101

Arithmetic manipulation:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost bit of the A field 1's complement.

1's complement	10001010	01011011
Initial carry		1
	10001010	01011010
Carries		11
2's complement	10001010	01011100

2. The B field contents are added to the 2's complement of the A field.

2's complement	10001010	01011100
B field	01000011	01101101
	11001001	00110001
Carries	1	11111
Final result	11001101	11001001

Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	10FB	10FC
B field contents (bin.)	11001101	11001001

The implied B address for a subsequent 4-byte command, following the execution of this SUBTRACT BINARY command, is 10FB (04347 dec.).

SUBTRACT SIGNED (PSUB)

Command code: 65 (41) (C1)

The SUBTRACT SIGNED command decimally subtracts the contents of the field specified by the effective A address from the contents of the field specified by the effective B address and stores the result in the B field. The initial contents of the A and B fields are assumed to be signed, packed (see PACK command description), decimal information with the sign stored in the rightmost four bits of each field. Fields containing negative values are indicated by the bit configuration 1101 in the sign position; any other configuration indicates the field to contain a positive value. The lengths of both the A and B fields are indicated by the T character, which may vary in value from 0 through 255, with 0 indicating a length of 256 bytes.

CAUTION
Do not subtract hexadecimal fields; the result is predictable, but not a correct hexadecimal difference.

Subtraction is performed using complementary addition, from right to left, one byte at a time, with a carry from one byte being added to the byte on its left. A carry beyond the leftmost byte is not included in the result; however, this carry sets the overflow flag ON. If a carry is not generated beyond the leftmost byte, the initial state of the overflow flag is preserved. The result of the subtraction replaces the original B field contents.

The result obtained when subtracting the contents of overlapping A and B fields may differ, depending upon the contents, from the result obtained when subtracting the contents of A and B fields that do not overlap (assuming that the A and B values are the same in both instances). The subtraction of overlapping fields normally changes the contents of the A field.

The sign of the result stored in the B field is determined by effectively changing the sign of the A field and then assuming the sign of the field with the greatest absolute value (algebraic sign derivation); if the absolute values of both the A and B fields are equal, the initial B field sign is retained. The appropriate bit configuration, 1011 (indicating a positive value) or 1101 (indicating a negative value) is stored in the B field if a sign change occurs.

The implied B address for a subsequent 4-byte command is the same as the effective B address available to the SUBTRACT SIGNED command prior to command execution.

The following examples show the three methods used by the SUBTRACT SIGNED command to determine the results of subtraction.

Example 1 (like signs with B field absolute value greater than A field absolute value)

Q	RA	A2	A1	T	RB	B2	B1
41	50	01	3B	03	78	00	A0

Partial B address = 00160 (dec.)
Index register 30, mode 0
Field length = 3 bytes
Partial A address = 00315 (dec.)
Index register 20, mode 0
Command code for 8-byte SUBTRACT SIGNED command

Assume:

Index register 20 contains OFFE (04080 dec.)
Index register 30 contains 10EO (04320 dec.)

After command setup:

Effective A address = 103B + OFFE = 1139 (04409 dec.)
Effective B address = 00A0 + 10EO = 1180 (04480 dec.)

Before command execution:

A field address (hex.)	1139		113A		113B	
A field contents (BCD)	0000	0100	0001	0101	0110	1011
Decimal equivalent	0	4	1	5	6	+

B field address (hex.)	1180		1181		1182	
B field contents (BCD)	0000	0111	0110	0101	0100	1011
Decimal equivalent	0	7	6	5	4	+

Arithmetic manipulation:

Decimal equivalent of subtraction to be performed: 07654(+) - 04156(+) = 03498(+)

BCD subtraction:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost digit of the A fields 1's complement.

1's complement	1111	1011	1110	1010	1001
Initial carry					1
	1111	1011	1110	1010	1000
Carries					1
2's complement	1111	1011	1110	1010	1010

2. The B field digits are added to the 2's complement of the A field.

2's complement	1111	1011	1110	1010	1010
B field digits	0000	0111	0110	0101	0100
	1111	1100	1000	1111	1110
Carries	1 1111	1111	11		
Uncorrected result	0000	0011	0100	1111	1110

3. A binary ten is added to the uncorrected result of any digit addition from step 1 and step 2 that did not generate a carry to the digit on its left (including the leftmost digit).

Uncorrected result	0000	0011	0100	1111	1110
Binary ten correction				1010	1010
	0000	0011	0100	0101	0100
Carries				1 11	1 11
Final result	0000	0011	0100	1001	1000

Carries between digits are ignored.

Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	1180		1181		1182	
B field contents (BCD)	0000	0011	0100	1001	1000	1011
Decimal equivalent	0	3	4	9	8	+

The implied B address for a 4-byte command following the execution of this SUBTRACT SIGNED command is 1180 (04480 dec.).

Example 2 (Like signs with A field absolute value greater than B field absolute value)

Assume that the fields used in the preceding example contain the following information before command execution:

A field address (hex.)	1139		113A		113B	
A field contents (BCD)	0000	0111	0110	0101	0100	1101
Decimal equivalent	0	7	6	5	4	-

B field address (hex.)	1180		1181		1182	
B field contents (BCD)	0000	0100	0001	0101	0110	1101
Decimal equivalent	0	4	1	5	6	-

Arithmetic manipulation:

Decimal equivalent of subtraction to be performed: 04156(-) - 07654(-) = 03498(+)

BCD subtraction:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost digit of the A fields 1's complement.

1's complement	1111	1000	1001	1010	1011
Initial carry					1
	1111	1000	1001	1010	1010
Carries					11
2's complement	1111	1000	1001	1010	1100

2. The B field digits are added to the 2's complement of the A field.

2's complement	1111	1000	1001	1010	1100
B field digits	0000	0100	0001	0101	0110
	1111	1100	1000	1111	1010
Carries			11	1111	1
Uncorrected result	1111	1100	1011	0000	0010

3. A binary ten is added to the uncorrected result of any digit addition from steps 1 and 2 that did not generate a carry to the digit on its left (including the leftmost digit).

Uncorrected result	1111	1100	1011	0000	0010
Binary ten correction	1010	1010	1010		
	0101	0110	0001	0000	0010
Carries	1 11	1	1 1		
Corrected result	1001	0110	0101	0000	0010

Carries between digits are ignored.

4. The 2's complement of the corrected result from step 3 is formed.

1's complement	0110	1001	1010	1111	1101
Initial carry					1
	0110	1001	1010	1111	1100
Carry					1
2's complement	0110	1001	1010	1111	1110

5. A binary ten is added to the 2's complement (from step 4) of any digit addition that did not generate a carry to the digit on its left (including the leftmost digit).

2's complement	0110	1001	1010	1111	1110
Binary ten correction	1010	1010	1010	1010	1010
	1100	0011	0000	0101	0100
Carries	1 11	1	1 1	1 11	1 11
Final result	0000	0011	0100	1001	1000

Carries between digits are ignored.

Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	1180		1181		1182	
B field contents (BCD)	0000	0011	0100	1001	1000	1011
Decimal equivalent	0	3	4	9	8	+

The implied B address for a subsequent 4-byte command, following the execution of this SUBTRACT SIGNED command, is 1180 (04480 dec.).

Example 3 (Unlike signs)

Assume that the fields used in example 1 contain the following information before command execution:

A field address (hex.)	1139		113A		113B	
A field contents (BCD)	0000	0111	0110	0101	0100	1011
Decimal equivalent	0	7	6	5	4	+

B field address (hex.)	1180		1181		1182	
B field contents (BCD)	0000	0001	0000	0001	0001	1101
Decimal equivalent	0	1	0	1	1	-

Arithmetic manipulation:

Decimal equivalent of subtraction to be performed: 01011(-) - 07654(+) = 08665(-)

BCD subtraction:

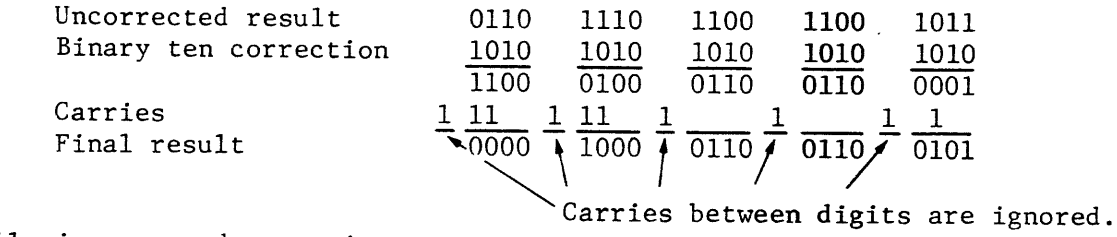
1. Excess six is added to each A field digit after it is read from memory.

A field digits	0000	0111	0110	0101	0100
Excess six	0110	0110	0110	0110	0110
	0110	0001	0000	0011	0010
Carries		11	11	1	1
Result	0110	1101	1100	1011	1010

2. The B field digits are added to the result obtained in step 1.

Result from step 1	0110	1101	1100	1011	1010
B field digits	0000	0001	0000	0001	0001
	0110	1100	1100	1010	1011
Carries		1		11	
Uncorrected result	0110	1110	1100	1100	1011

3. A binary ten is added to the uncorrected result of any digit addition (from step 2) which did not generate a carry to the digit on its left (including the leftmost digit).



Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	1180		1181		1182	
B field contents (BCD)	0000	1000	0110	0110	0101	1011
Decimal equivalent	0	8	6	6	5	+

The implied B address for a subsequent 4-byte command, following the execution of this SUBTRACT SIGNED command, is 1180 (04480 dec.).

SUBTRACT UNSIGNED (USUB)

Command code: 99 (63) (E3)

The SUBTRACT UNSIGNED command decimally subtracts the contents of the field specified by the effective A address from the contents of the field specified by the effective B address and places the result in the B field. The initial contents of the A and B fields are assumed to be ASCII characters 0 through 9. The lengths of both the A and B fields are indicated by the T character, which may vary in value from 0 through 255, with 0 indicating a length of 256 bytes.

CAUTION
Do not subtract hexadecimal fields; the result is predictable, but not a correct hexadecimal difference.

Subtraction is performed (using complementary addition) from right to left, one byte at a time, operating on only the rightmost four bits of each byte; the leftmost four bits are ignored. A carry beyond the rightmost four bits of one byte are added to the byte on its left. A carry beyond the leftmost byte is discarded. The result of the subtraction is stored in the rightmost four bits of each B field byte; the ASCII bit configuration 0011 is stored in the leftmost four bits. A negative result is stored as the tens complement of the B field value subtracted from the A field value, and the overflow flag is set ON.

The result obtained from subtracting the contents of overlapping A and B fields may differ from the result obtained when subtracting the contents of A and B fields that do not overlap (assuming the A and B field values are the same in both instances). The subtraction of overlapping fields normally changes the contents of the A field.

The implied B address for a subsequent 4-byte command is the same as the

effective B address established prior to execution of the SUBTRACT UNSIGNED command.

Example 1 (A field value less than B field value)

Q	RA	A2	A1	T	RB	B2	B1
63	90	02	00	03	B0	00	00

Partial B address = 00000 (dec.)
Index register 44, mode 0
Field length = 3 bytes
Partial A address = 00512 (dec.)
Index register 36, mode 0
Command code for 8-byte SUBTRACT UNSIGNED command

Assume:

Index register 36 contains 0E10 (03600 dec.)
Index register 44 contains 1130 (04400 dec.)

After command setup:

Effective A address = 0200 + 0E10 = 1010 (04112 dec.)
Effective B address = 0000 + 1130 = 1130 (04400 dec.)

Before command execution:

A field address (hex.)	1010	1011	1012
A field contents (ASCII)	0011 0100	0011 0100	0011 0010
Equivalent	4	4	2

B field address (hex.)	1130	1131	1132
B field contents (ASCII)	0011 0111	0011 0010	0011 0011
Equivalent	7	2	3

Arithmetic manipulation:

Decimal equivalent of subtraction to be performed:

```
  723
 -442
 ---
  281
```

BCD subtraction:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost digit of the A field 1's complement.

1's complement	1011	1011	1101
Initial carry			1
	1011	1011	1100
Carry			1
2's complement	1011	1011	1110

2. The 2's complement of the A field is added to the B field.

2's complement	1011	1011	1110
B field digits	0111	0010	0011
	1100	1001	1101
Carries	1 111	111	11
Uncorrected result	0010	1110	0001

3. A binary ten is added to the uncorrected result of any digit addition from step 1 or 2 that did not generate a carry to the digit on its left (including the leftmost digit).

Uncorrected result	0010	1110	0001
Binary ten correction		1010	
	0010	0100	0001
Carries	1	11	
Final result	0010	1000	0001

Carries between digits are ignored.

4. The results from step 3 are stored in b4 - b1 of the corresponding B field bytes, and b8 - b5 of each byte are set to 0011.

Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	1130		1131		1132	
B field contents (ASCII)	0011	0010	0011	1000	0011	0001
Equivalent	2		8		1	

The implied B address for a subsequent 4-byte command, following the execution of this SUBTRACT UNSIGNED command, is 1130 (04400 dec.).

Example 2 (A field value greater than B field value)

Assume that the fields used in the preceding example contain the following information before command execution:

A field address (hex.)	1010		1011		1012	
A field contents (ASCII)	0011	0111	0011	0010	0011	0011
Equivalent	7		2		3	

B field address (hex.)	1130		1131		1132	
B field contents (ASCII)	0011	0100	0011	0100	0011	0010
Equivalent	4		4		2	

Arithmetic manipulation:

Decimal equivalent of subtraction to be performed:

```
 442
 723
-281
```

Since the result is negative, the tens complement of the A field value minus the B field value is stored.

```
  1000
- 281
  719   Tens complement
```

BCD subtraction:

1. The 2's complement of the A field is formed by adding an initial carry to the rightmost digit of the A field 1's complement.

1's complement	1000	1101	1100
Initial carry			1
2's complement	1000	1101	1101

2. The 2's complement of each A field digit is added to the corresponding B field digit.

2's complement	1000	1101	1101
B field digits	0100	0100	0010
	1100	1001	1111
Carries	1	1	
Uncorrected result	1101	0001	1111

3. A binary ten is added to the uncorrected result of any digit addition from step 1 or 2 that did not generate a carry to the digit on its left (including the leftmost digit).

Uncorrected result	1101	0001	1111
Binary ten correction	1010		1010
	0111	0001	0101
Carries	1		1 11
Tens complement of result	0111	0001	1001

Carries between digits are ignored.

4. The tens complement of the result is stored in b4 - b1 of the corresponding B field digit, and b8 - b5 of each byte are set to 0011.

Following command execution:

The A field contents are unchanged.

B field address (hex.)	1130		1131		1132	
B field contents (ASCII)	0011	0111	0011	0001	0011	1001
Equivalent	7		1		9	

The overflow flag is ON, and the implied B address for a subsequent 4-byte command, following the execution of this SUBTRACT UNSIGNED command, is 1130 (04400 dec.).

TEST BIT (TESTB)

Command code: 83 (53) (D3)

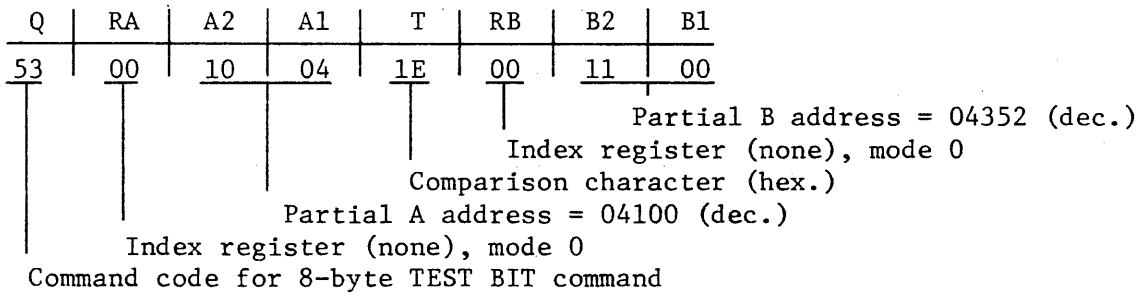
The TEST BIT command transfers program control to either the command indicated by the effective A address or to the next command in sequence, as determined by the bit configuration of both the T character value and the 1-byte field specified by the effective B address. The effective A address must be 0 mod 4

(evenly divisible by 4) or a program error occurs, leaving the initial contents of the control register unchanged.

Effectively, the T character bit positions (bit eight through bit one) with a value of one are compared to the corresponding B field bit positions; if these T character bits are matched by corresponding B field bits with a value of one or if all the T character bits are zero, program control is transferred to the command indicated by the effective A address and the repeat indicator is set OFF. If any of the T character bits with a value of one are matched by corresponding B field bits with a value of zero, program control is transferred to the next command in sequence, and the repeat indicator is undisturbed.

The implied B address for a subsequent 4-byte command, following the execution of the TEST BIT command, is the same as the effective B address available prior to the execution of this TEST BIT command.

Example



After command setup:

Effective A address = 1004 (04100 dec.)
Effective B address = 1100 (04352 dec.)

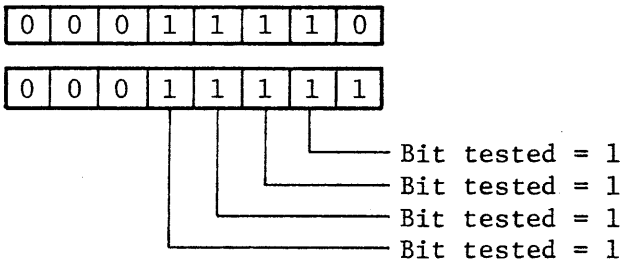
Before command execution:

The contents of the A field are not significant.

B field address (hex.)	1100	
B field contents	0001	1111
Hexadecimal equivalent	1	F

During command execution:

The bit configurations of the T character and B field are tested as shown below.



Following command execution:

Since the T and B bits tested are equal, the repeat indicator is set OFF and program control is transferred to the command specified by the effective A address (1004 hex.).

The implied B address for a subsequent 4-byte command following the execution of this TEST BIT command is 1100 (04352 dec.).

TEST CHARACTER EQUAL (TESTCE)

Command code: 81 (51) (D1)

The TEST CHARACTER EQUAL command binarily compares the bit configuration of a 1-byte field specified by the effective B address to the bit configuration of the T character and, depending on the result, transfers program control to either the command indicated by the effective A address or to the next command in sequence. The effective A address must be 0 mod 4 (evenly divisible by four) or a program error occurs, leaving the initial contents of the control register undisturbed.

Transfer of program control is determined as follows: If the bit configurations of the T character and B field are identical, program control is transferred to the command indicated by the effective A address, and the repeat indicator is set OFF; if the bit configurations are not identical, program control is transferred to the next command in sequence and the repeat indicator remains unchanged. (The G, E, and L flags are not used by this command.)

The implied B address available to a subsequent 4-byte command is the same as the effective B address established prior to the execution of the TEST CHARACTER EQUAL command.

Example

Q	RA	A2	A1	T	RB	B2	B1
51	00	10	F8	2F	00	0F	FB

Partial B address = 04080 (dec.)
Index register (none), mode 0
Comparison character (hex.)
Partial A address = 04344 (dec.)
Index register (none), mode 0
Command code for 8-byte TEST CHARACTER EQUAL command

After command setup:

Effective A address = 10F8 (04344 dec.)
Effective B address = 0FFB (04080 dec.)

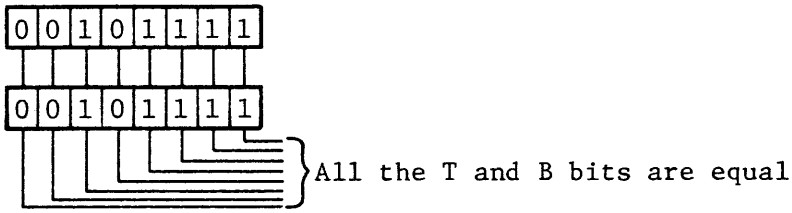
Before command execution:

The contents of the A field are not significant.

B field address (hex.)	OFFB	
B field contents (bin.)	0010	1111
Hexadecimal equivalent	2	F

During command execution:

The bit configuration of the T character and B field are tested as shown below.



Following command execution:

Since all the T and B bits are equal, the repeat indicator is set OFF and program control is transferred to the command specified by the effective A address (10F8 hex.).

The implied B address for a subsequent 4-byte command following the execution of this TEST CHARACTER EQUAL command is OFFB (04080 dec.).

TEST CHARACTER UNEQUAL (TESTCU)

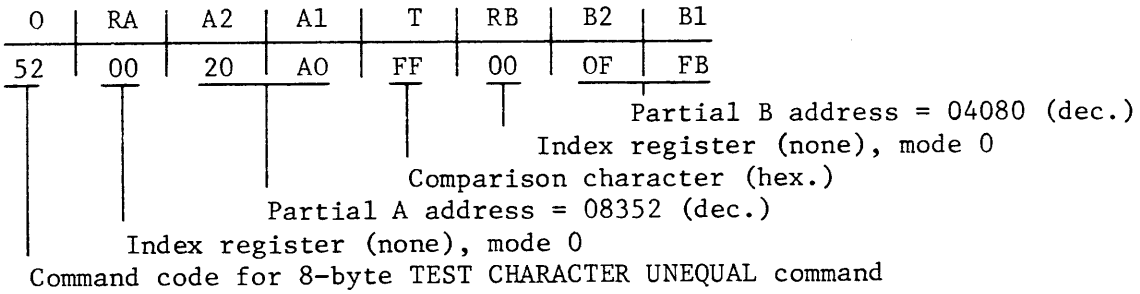
Command code: 82 (52) (D2)

The TEST CHARACTER UNEQUAL command, binarily compares the T character bit configuration with the bit configuration of the 1-byte field specified by the effective B address and, depending upon the result, transfers program control to either the command indicated by the effective A address or to the next command in sequence. The effective A address must be 0 mod 4 (evenly divisible by four) or a program error occurs, leaving the initial contents of the control register undisturbed.

Transfer of program control is determined as follows: If the bit configurations of the T character and B field are not identical, program control is transferred to the command specified by the effective A address and the repeat indicator is set OFF; if the bit configurations are identical, program control is transferred to the next command in sequence and the initial state of the repeat indicator is unchanged. (The G, E, and L flags are not used by this command.)

The implied B address available to a subsequent 4-byte command is the same as the effective B address of the TEST CHARACTER UNEQUAL command established prior to command execution.

Example



After command setup:

Effective A address = 20A0 (08352 dec.)
Effective B address = OFFB (04080 dec.)

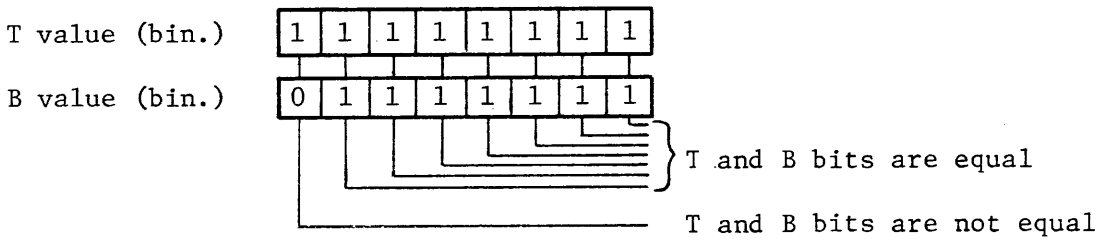
Before command execution:

The contents of the A field are not significant

B field address (hex.)	OFFB	
B field contents (bin.)	0111	1111
Hexadecimal equivalent	7	F

During command execution:

The bit configurations of the T character and B field are tested as shown below.



Following command execution:

Since the T and corresponding B bits are not all equal, the repeat indicator is set OFF and program control is transferred to the command specified by the effective A address (20A0 hex.).

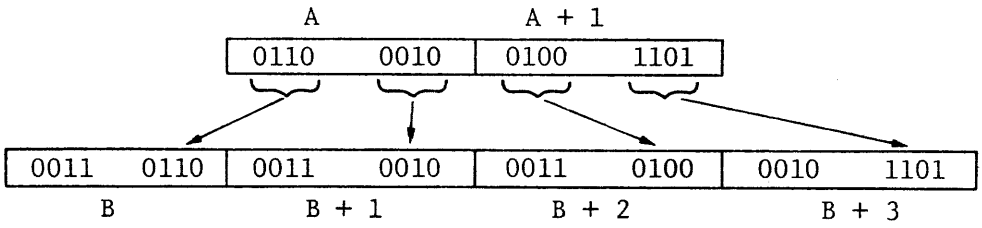
The implied B address for a subsequent 4-byte command, following the execution of this TEST CHARACTER UNEQUAL command, is OFFB (04080 dec.).

UNPACK (UNPACK)

Command code: 77 (4D) (CD)

The UNPACK command restores the contents (see PACK command description) of the field specified by the effective A address to the appropriate ASCII characters and places the result in the field specified by the effective B address. The length of the A field is indicated by the T character, which may vary in value from 0 through 255, with 0 indicating a length of 256 bytes. The length of the B field is equal to twice the length of the A field.

Unpacking is performed sequentially from left to right, starting with the leftmost A and B field memory locations. The first A field byte is separated into two 4-bit digits with the leftmost four bits replacing the rightmost bits of the first B field byte; the rightmost four bits of the A field byte replace the rightmost four bits of the second B field byte. The leftmost four bits of each B field byte are set to the appropriate ASCII zone bit configuration (0010 or 0011). If the new binary value of the rightmost four bits of the B field byte is greater than nine (a positive or negative sign configuration), the 0010 configuration is used; if the value is nine or less, the 0011 configuration is used. This process is repeated until all the A field contents have been unpacked and placed in the B field.



On completion of the command, the A field contains its initial contents, and the B field contains the unpacked data.

CAUTION

The result obtained when using the UNPACK command with overlapping A and B fields may differ from the result obtained when using it with fields that do not overlap (assuming the A and B fields contents are the same in both instances). The contents of the A field normally are changed when the fields overlap.

The implied B address for a subsequent 4-byte command is equal to the sum of adding twice the length of the A field to the effective B address established prior to the execution of the UNPACK command.

Example 1

Q	RA	A2	A1	T	RB	B2	B1
4D	40	OD	00	02	BO	02	58

Partial B address = 00600 (dec.)
Index register 44, mode 0
Length of A field = 2 bytes
Partial A address = 03328 (dec.)
Index register 16, mode 0
Command code for 8-byte UNPACK command

Assume:

Index register 16 contains 0640 (01600 dec.)
Index register 44 contains 1130 (04400 dec.)

After command setup:

Effective A address = ODOO + 0640 = 1340 (04982 dec.)
Effective B address = 0258 + 1130 = 1388 (05000 dec.)

Before command execution:

A field address (hex.)	1340		1341	
A field contents	0000	1000	0100	1011
Equivalent	0	8	4	+

The contents of the B field are not significant.

Following command execution:

The contents of the A field are unchanged.

B field address (hex.)	1388	1389	138A	138B
B field contents (ASCII)	0011 0000	0011 1000	0011 0100	0011 1011
Equivalent	0	8	4	+

The implied B address for a subsequent 4-byte command, following the execution of this UNPACK command, is 138C (05004 dec.).

Example 2 (Fields overlap)

Q	RA	A2	A1	T	RB	B2	B1
4D	34	01	A6	02	00	11	33

Partial B address = 04403 (dec.)
Index register (none), mode 0
Length of A field = 2 bytes
Partial A address = 00422 (dec.)
Index register 13, mode 0
Command code for 8-byte UNPACK command

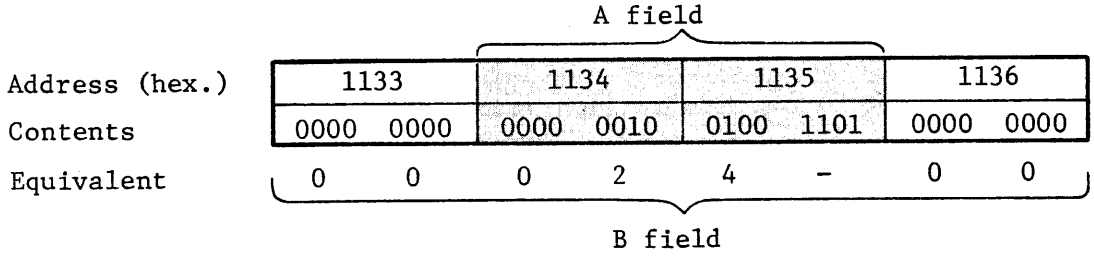
Assume:

Index register 13 contains OF8E (03982 dec.)

After command setup:

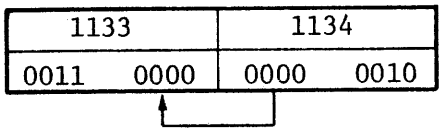
Effective A address = 01A6 + OF8E = 1134 (04404 dec.)
Effective B address = 1133 (04403 dec.)

Before command execution:

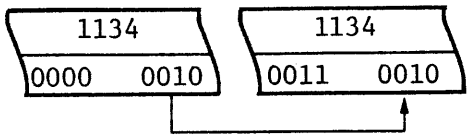


During command execution:

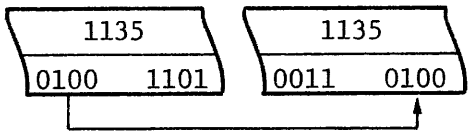
1. B8 - b5 of address 1134 replaces b4 - b1 of address 1133; zone bits 0011 are appended to address 1133.



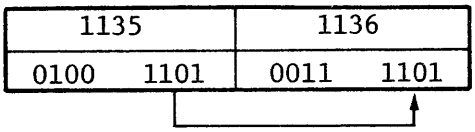
2. B4 - b1 of address 1133 remain unchanged since they replace themselves. Zone bits 0011 replace b8 - b5 of address 1134.



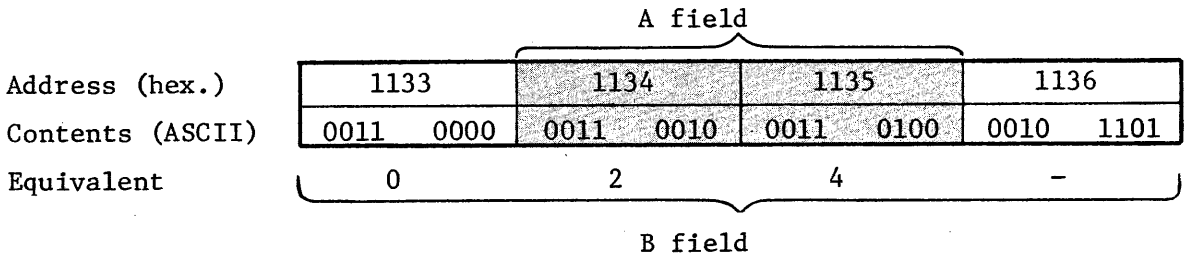
3. B8 - b5 of address 1135 replace b4 - b1 of address 1135. Zone bits 0011 replace b8 - b5 of address 1135.



4. B4 - b1 of address 1135 replace b4 - b1 of address 1136. When the contents of address 1135 were originally setup for unpacking, all eight bits were moved to a temporary storage area and unpacked from there so that the original b4 - b1 bits (1101) of address 1135, not the bits obtained in step 3 (0100), replace b4 - b1 of address 1136.



Following command execution:



The implied B address for a subsequent 4-byte command, following the execution of this UNPACK command, is 1137 (04407 dec.).

WAIT (WAITI)

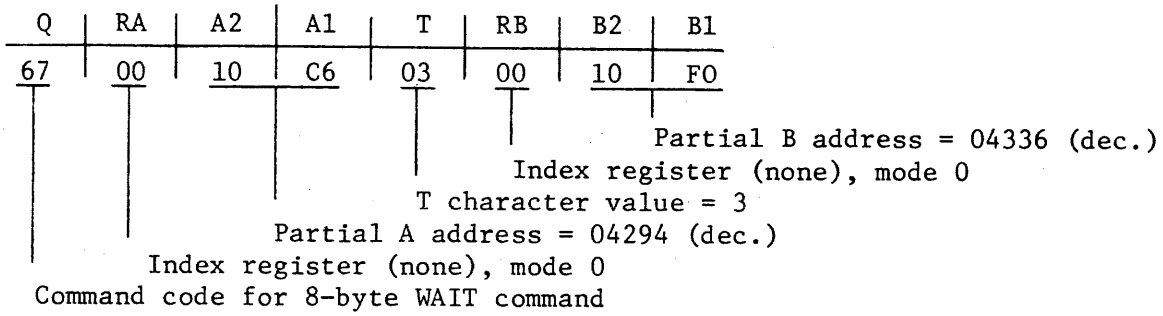
Command code: 103 (67) (E7)

The WAIT command sets the WAIT light ON, sets the repeat indicator OFF, and displays its effective A address in the ADDRESS DISPLAY lights located on the Operator's console; it then enters a loop by decrementing the address contained in the control register by either 4 or 8 depending upon whether a 4 or 8-byte command is being executed. Between-commands testing takes place between each execution of the WAIT command, during the loop. Input and output (I/O) operations which are being processed during the first execution of the WAIT command are allowed to continue until completion. I/O termination interrupt is permitted (refer to PRODUCT INFORMATION REFERENCE MANUAL, PROCESSORS, "NCR Century 101 Processor," under the heading of Termination of I/O Operation).

The WAIT command continues looping until either the HALT switch is set ON or the COMPUTE switch is pressed (both switches are located on the Operator's console). If the HALT switch is set ON, looping terminates and the processor enters the halt state. If the COMPUTE switch is pressed, looping terminates, the WAIT light is set OFF, and the processor transfers program control to the next command in sequence; the implied B address available to a subsequent

4-byte command is identical to the effective B address established prior to the execution of the WAIT command.

Example



Assume:

The compute indicator is OFF.

After command setup:

Effective A address = 10C6 (04294 dec.)
Effective B address = 10F0 (04336 dec.)

Following command execution:

The WAIT command begins the set up process again.

The WAIT DISPLAY indicator is ON, the repeat indicator OFF, and the effective A address is displayed through the ADDRESS DISPLAY lights on the Operator's console.

The implied B address for a subsequent 4-byte command is 10F0 (04336 dec.).

COMMAND TIMING

Total command time is calculated as the sum of the setup time (including indirect addressing and incremental indexing) plus the command execution time:

$$C = S + E$$

Where:

C = Total time in microseconds required for command setup and execution.

S = Total time in microseconds required for command setup.

E = Total time in microseconds required for command execution.

COMMAND SETUP TIMING

Four equations are available to determine the time reuqired to setup (S) a particular command:

$$TA = 1.2\ [3(MZN) + 5(MZR) + 9(M2) + 6(M3)]$$

$$TB = 1.2\ [3(MZN) + 5(MZR) + 9(M2) + 6(M3)]$$

$$T1A = 1.2\ [8 + 5(N - 1) + 2(MZN) + 2(MZR) + 6(M2) + 3(M3)]$$

$$T1B = 1.2\ [8 + 5(N - 1) + 2(MZN) + 2(MZR) + 6(M2) + 3(M3)]$$

Where:

TA is the setup time required for the A operand, using mode zero without an index register, mode zero with an index register, mode two, or mode three addressing.

TB is the setup time required for the B operand, using mode zero without an index register, mode zero with an index register, mode two or mode three addressing.

T1A is the setup time required for the A operand, using mode one addressing (including different levels) as well as the type of addressing used to terminate the flow.

T1B is the setup time required for the B operand, using mode one addressing (including different levels) as well as the type of addressing used to terminate the flow.

The variables used in the preceding equations assume the following values:

MZN = 1, if mode zero addressing is used without an associated index register; otherwise, it is equal to zero.

MZR = 1, if mode zero addressing is used with an associated index register; otherwise, it is equal to zero.

M2 = 1, if mode two addressing is used; otherwise, it is equal to zero.

M3 = 1, if mode three addressing is used; otherwise, it is equal to zero.

N = the number of levels of mode one addressing used (not including the mode of addressing used at termination).

The sum of the appropriate equations (TA, TB, T1A, and T1B) are used to arrive at the setup time for any 8-byte command; however, only TA or T1A is used to determine the 4-byte command setup time.

Example

Assume that the setup time is to be determined for an 8-byte command, using mode zero addressing with an index register for the A operand, and two levels of mode one addressing, terminating with mode three addressing for the B operand.

1. The setup time for the A operand is determined as follows:

   TA = 1.2 [3(MZN) + 5(MZR) + 9(M2) + 6(M3)]

   TA = 1.2 [3(0) + 5(1) + 9(0) + 6(0)] = 6

2. The setup time for the B operand is determined as follows:

   T1B = 1.2 [8 + 5(N - 1) + 2(MZN) + 2(MZR) + 6(M2) + 3(M3)]

   T1B = 1.2 [8 + 5(2 - 1) + 2(0) + 2(0) + 6(0) + 3(1)] = 19.2

3. The setup time for the entire command is:

   S = 6 + 19.2 = 25.2 microseconds

COMMAND EXECUTION TIMING

The following table contains the equations used to determine the execution time (when variable) for each command and flow (excluding the I/O interrupt flow), or the actual execution time in microseconds. The variable (T) is used to specify the number of bytes on which the command or flow is operating.

COMMAND OR FLOW	EXECUTION TIME
ADD BINARY	E = 1.2[6 + 3(T - 1)]
ADD SIGNED	E = 1.2[9 + 3(T - 1)]
ADD SIGNED *	E = 1.2[17 + 6(T - 1)]
ADD UNSIGNED	E = 1.2[6 + 3(T - 1)]
BRANCH (ALL BRANCH COMMANDS)	E = 2.4
COMMAND CODE TRAP FLOW	E = 16.8
COMPARE BINARY	E = 1.2[6 + 3(T - 1)]
COMPARE SIGNED	E = 1.2[9 + 3(T - 1)]
DECODE ALL	E = 1.2[8 + 5(T - 1)]
DECODE TO DELIMITER	E = 1.2[8 + 5(T - 1)]
DIVIDE	E = 49 MIN, 6800 MAX
INOUT	E = 10.8 MIN
INTERRUPT FLOW	E = 14.4
JUMP	E = 3.6
LOGIC	E = 4.8
MEMORY ERROR FLOW	E = 16.8
MOVE A LEFT TO RIGHT	E = 1.2[3 + 2(T - 1)]
MOVE A RIGHT TO LEFT	E = 1.2[5 + 2(T - 1)]
MOVE B RIGHT TO LEFT	E = 1.2[5 + 2(T - 1)]
MULTIPLY	E = 16.8 MIN, 798 MAX
OPTION SWITCHES INPUT	E = 0
PACK	E = 1.2[4 + 3(T - 1)]
PROGRAM ERROR FLOW	E = 16.8
REPEAT	E = 3.6
REPEAT **	E = 7.2
REPEAT FLOW	E = 4.8
RESTORE	E = 9.6
SET IP OFF	E = 2.4
SET IP ON	E = 2.4
SUBTRACT BINARY	E = 1.2[6 + 3(T - 1)]
SUBTRACT SIGNED	E = 1.2[9 + 3(T - 1)]
SUBTRACT SIGNED *	E = 1.2[17 + 6(T - 1)]
SUBTRACT UNSIGNED	E = 1.2[6 + 3(T - 1)]
TEST BIT	E = 3.6
TEST CHARACTER EQUAL	E = 3.6
TEST CHARACTER UNEQUAL	E = 3.6
UNPACK	E = 1.2[4 + 3(T - 1)]
WAIT	E = 2.4
* If recomplementing is necessary ** If the repeat number is zero	

76.12

AN EDUCATIONAL PUBLICATION

STOCK NO. ST-9402-04 **CHANGE NO.**_____

NCR REFERENCE LIBRARY

NCR CENTURY PRODUCT INFORMATION

RM-0141

PLACE IN REFERENCE BINDER

INSERT	DELETE	MAJOR SECTION / MINOR SECTION / MODULE TITLE	PUB REF	PAGES	DATE	COMMENT
	X	PROCESSORS	5.2	ALL	Jun. 74	
X		PROCESSORS	5.2	1-2	Jan. 77	The attached revision reflects increased trunk bandwidths and maximum memory size in the NCR Century 151 processors.

NCR REFERENCE MANUAL

An Educational Publication

5.2
1 of 2
Jan. 77

ST-9402-04
BINDER NO. 0141

NCR CENTURY 151 PROCESSOR

This publication contains descriptions of the differences, involving memory, I/O control, and other enhancements, between the NCR Century 151 and the NCR Century 101 processors. The NCR Century 101 Product Information publications should be used in conjunction with this publication for a complete description of the structure and functions of the NCR Century 151.

GENERAL DESCRIPTION

The NCR Century 151 processor is basically similar in appearance and function to the NCR Century 101 processor, but has nearly twice the speed. Because of the greater speed of operation, I/O trunk bandwidths have been significantly increased. The NCR Century 151 also offers certain other features, such as I/O control line parity and microdiagnostic testing facilities, which, while not apparent to the user, are of great assistance in maintaining the functional status of the system. These basic differences are described in greater detail below and on the following page.

MEMORY

The memory of the NCR Century 151 distinguishes it from all other members of the NCR Century family of computer systems. The NCR Century 151 uses MOS (Metallic-Oxide Semiconductor) memory, which is almost twice as fast as the conventional ferrite core memory of the NCR Century 101. The NCR Century 151 has a memory cycle time of 0.75 microseconds, as compared with the 1.2 microsecond cycle time of the NCR Century 101.

The minimum memory size of the NCR Century 151 processor is 64K bytes, which can be expanded to 128K bytes in 32K increments, then in 64K increments to a maximum size of 256K (K = 1024). The Extended Addressing Logic (EAL) feature provides the proper addressing scheme for memory sizes greater than 64K.

I/O CONTROL

Two significant advances in the area of Input/Output operations are offered by the NCR Century 151 processor, as described below.

Trunk Bankwidths

Because of the rapid cycle time, the bandwidths of the high-speed I/O trunks 6 and 7 are greatly increased. In addition, trunk 6 is double-buffered, further enhancing its bandwidth. I/O trunk bandwidths in the NCR Century 151 are shown in the following table.

Trunk	Bandwidth
0	240 KB
1	266 KB
6	1333 KB
7	666 KB

Trunks 1 and 6, which are optional with the NCR Century 101, are included in the basic NCR Century 151 system.

Control Line Parity

All control characters are accompanied by an odd parity bit that is generated by the transmitting unit and checked by the receiving unit, to ensure accuracy in transmitting the control characters and to guard against selecting a wrong peripheral unit. Parity bits are used with control characters in the selection of a peripheral, acknowledgement and service requests by the peripheral, I/O terminating signals by either the I/O Control or the peripheral, and transmission of status characters to indicate the outcome of the I/O operation.

DIAGNOSTIC AND TESTING FACILITIES

The NCR Century 151 provides diagnostic and testing facilities of three types:

- Malfunctions are detected and logged to aid the field engineer in locating patterns or trends in the occurrence of those malfunctions. This disc log is valuable in isolating intermittent faults and those faults that do not disable the system.

- Diagnostic programs, written in an elementary machine language that uses microdiagnostic hardware (nonprogrammable wired-in logic), are used to isolate faults in the ALU and I/O Control where extensive testing at internal speeds is required. Microdiagnostics are extremely valuable in establishing the integrity of the basic processing system.

- Comprehensive test panels are included in the individual units (ALU, I/O Control, memory, and peripherals) for effective offline testing and troubleshooting of solid or predictable faults.

SYSTEM PERIPHERALS

The peripherals commonly used with the NCR Century 151 are listed in the "Peripheral Transfer Rates and System Configurations" publication in the System Installation section of this binder.

OPTIONS

The options available with the NCR Century 151 are essentially the same as those offered with the NCR Century 101, except that the Extended Addressing logic (EAL) feature and common trunk 1 and buffered trunk 6 are not options, but are included in the basic NCR Century 151 system.

NCR REFERENCE MANUAL

An Educational Publication

number: 5.3
page: 1 of 1
date: Sep. 76

ST-9402-03
BINDER NO. 0141

NCR CENTURY 75 PROCESSOR

This publication contains descriptions of the differences, involving memory, standard system peripherals, and options, between the NCR Century 75 and the NCR Century 101 processors. The NCR Century 101 Product Information publication should be used in conjunction with this publication for a complete description of the structure and functions of the NCR Century 75.

GENERAL DESCRIPTION

The NCR Century 75 is identical in appearance to the NCR Century 101 and has the same internal performance characteristics and instruction set, although it is limited to two I/O trunks and has fewer available peripherals. It is, however, considerably lower in price. As with other members of the NCR Century family of computers, the NCR Century 75 is upward-compatible. The basic system can be easily upgraded with faster printers, additional discs, and more memory. Should further upgrading be desirable, the NCR Century 75 can be converted, on-site, to an NCR Century 101.

MEMORY

The basic NCR Century 75 processor is equipped with 16K bytes of ferrite-core memory. This can be expanded in 8K increments to 32K, then in 16K increments to the maximum memory size of 64K bytes (K = 1024).

SYSTEM PERIPHERALS

The standard system peripherals on the NCR Century 75 are basically the same as those on the NCR Century, with two exceptions:

- The printer in the basic NCR Century 75 system is the 640-122 integrated printer (200 lines per minute), instead of the 649-300 freestanding printer. The 640-132 (300 LPM) and 640-102 (450 LPM) printers are available as substitutions. The 649-300 printer is not available for use with the NCR Century 75.

- The basic NCR Century 75 is equipped with the Teletype I/O Writer instead of the thermal-printing I/O Writer.

OPTIONS

The Extended Addressing Logic (EAL) feature and the Multiprogramming feature are not available as options to the NCR Century 75, nor are the additional common trunks 1 and 6. All other options offered to the NCR Century 101 may be added to the NCR Century 75.