NCR REFERENCE MANUAL

An Educational Publication

PRODUCT INFORMATION--NCR CENTURY PROCESSORS

number: 2
page: 1 of 68
date: Feb. 70

ST-9402-11
BINDER NO. 0141

# NCR CENTURY 100 PROCESSOR

67-1639-9

This publication is a general functional description of NCR Century 100 processor operations. While the information it contains is accurate, the material has been somewhat simplified for a specific audience. The publication is not intended as a programming or operating manual, and should not be construed as such.

# TABLE OF CONTENTS

## INTRODUCTION

## MEMORY

## ARITHMETIC LOGIC UNIT

## I/O CONTROL

## OPERATOR'S CONSOLE AND GENERAL OPERATING PROCEDURES

OPTIONAL I/O WRITER

NCR CENTURY 100 SPECIFICATIONS

# INTRODUCTION

## THE NCR CENTURY 100 PROCESSOR

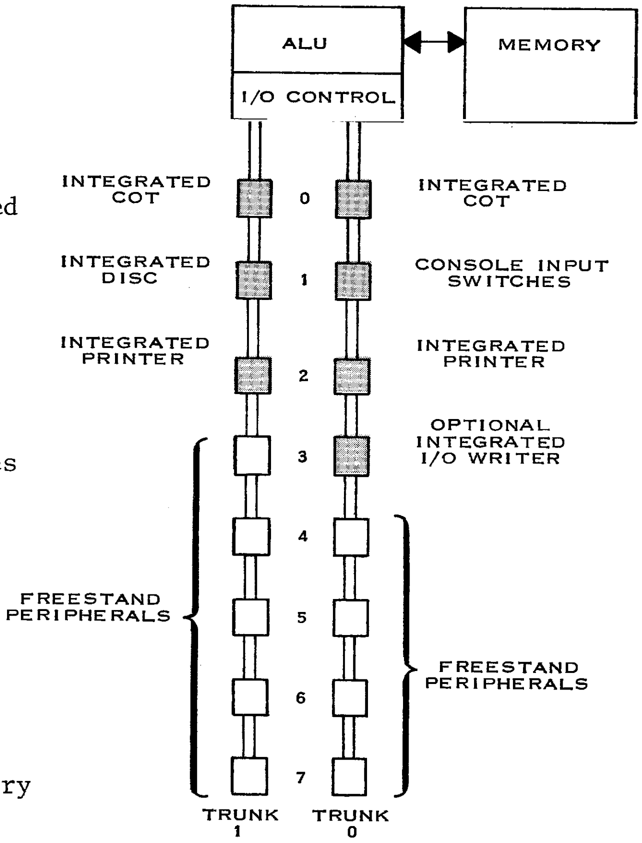
The NCR Century 100 processor is the basic member of the NCR Century Series. It is designed principally for sequential-program (batch) processing and certain online (real-time) processing. The NCR Century 100 processor includes:

- a common trunk for peripherals
- binary addressing
- 63 index registers
- industry-standard 8-bit character representation
- character addressing
- simple command structure
- 4- or 8-character command structure with implied T and B for 4-character structure

## THE NCR CENTURY 100 SYSTEM

The basic NCR Century 100 System consists of a central processor, high-speed memory, a dual-spindle disc unit, a printer, and a punched card or punched tape reader. The disc unit, printer, and card or tape reader are integrated peripherals; that is, all the logic required for controlling these units is an integral part of the central processor.

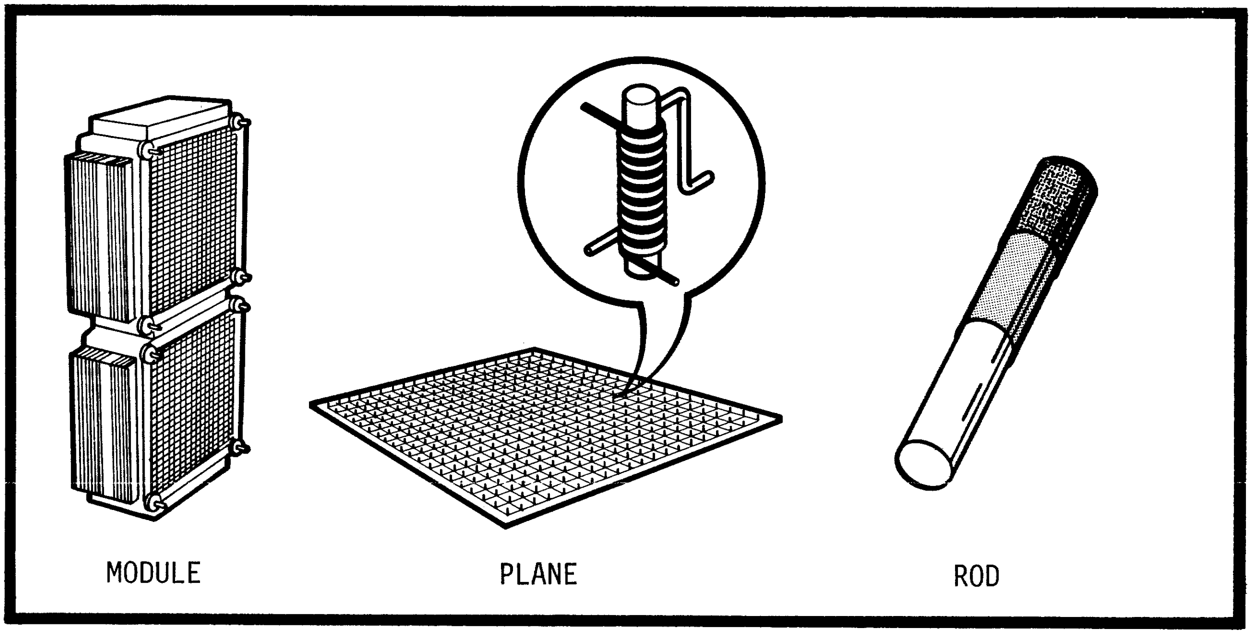
The processor comprises an arithmetic logic unit (ALU) and an input/output control unit (I/O Control). The I/O Control contains the logic circuitry required to handle communication between the processor and each of its peripherals. This communication takes place over two lines, designated Trunk 0 and Trunk 1 in the illustration at the right. Each trunk can handle eight peripheral units, with certain positions reserved exclusively for integrated peripherals and the remainder available for freestanding units. The NCR Century common trunk concept specifies a standard interface logic that permits the processor to communicate with NCR Century peripherals, regardless of type or function, through any open position on either trunk. The I/O Control section of this publication contains a detailed explanation of NCR Century processor/peripheral communication. For a complete description of the NCR Century 100 cabling, refer to the Site Preparation publication in this series.



# MEMORY

## PHYSICAL DESCRIPTION

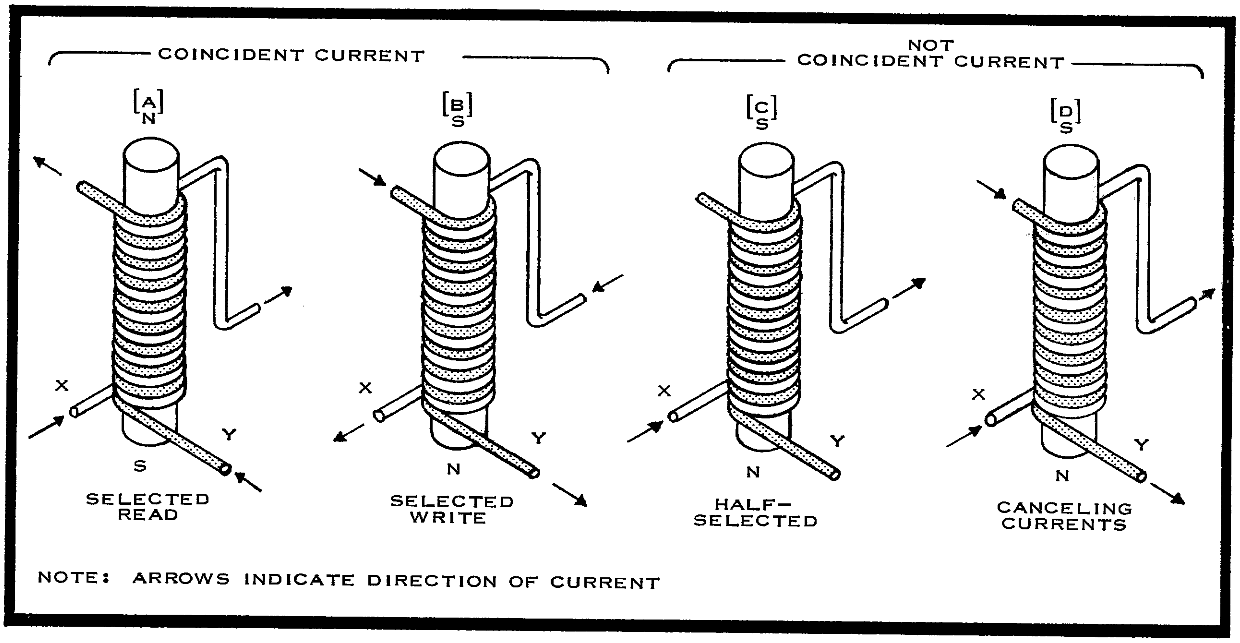
The basic memory element in all NCR Century Series computers is a rod .110 inches long and .006 inches in diameter. Rods are made by depositing a thin film of nickel-iron and a protective urethane coating on a length of beryllium-copper wire. The basic memory plane, containing 4608 rods, is formed by inserting the individual bit rods into solenoid coils wound on a plastic frame. The entire plane is then sealed between two protective plastic sheets. Planes in turn are wired together to from stacks (16 planes/stack). Two stacks constitute an NCR Century memory module capable of storing 16,384 characters of information. The basic NCR Century 100 System has one memory module with a second module optional, increasing memory capacity to 32,768 characters.



## Short-rod Storage

Each rod stores one bit (binary digit) by being magnetized in one of two directions (indicated by 1 or 0). Eight bits, considered as a single character of information, constitute a byte, the smallest addressable unit of data in the system. A ninth bit is appended to each byte to function as a parity check, but is not accessible to the program. The NCR Century Series makes an odd parity check on transferred data. This means that if the first eight bits of a byte contain an even number of 1's, the ninth - or parity check-bit is set to 1 so that the total is an odd number. If the first eight bits contain an odd number of 1's, the parity check bit should be 0. NCR Century hardware generates a parity bit when data is written into memory and checks parity when data is read out of memory. An error interrupt (ME) occurs when a parity error is detected.

The illustration below shows four drawings of a rod and its associated windings to demonstrate the magnetic effects of current flow through memory:



NOTE: ARROWS INDICATE DIRECTION OF CURRENT

The windings consist of two electrically insulated wires interwoven around the rod in the same direction. Each winding has ten turns; the total magnetic strength of the current is the algebraic sum of the magnetic strength for each wire. Assume for purposes of illustration that the current in each wire is sufficient to produce half the magnetic strength required to reverse the state of the rod (in other words, half the magnetic strength required to change the rod's bit representation from 0 to 1 or 1 to 0).

A. The rod assumes the north (N) and south (S) poles as indicated. The rod is said to be selected since the current is coincident; that is, current in both wires is flowing in the proper direction to establish a magnetic state. Coincident current flows in the read direction in this example. (See Principles of Reading and Writing, below).

B. The rod is selected with coincident write current. Assume the current has been raised from 0 to maximum in the direction shown. Notice that the current flows in the opposite direction from current in example A. As current approaches maximum, the magnetic polarity of the rod changes (N and S reverse) because the magnetic field developed by the windings is induced in the rod. This reversal of magnetic field is referred to as flipping the rod.

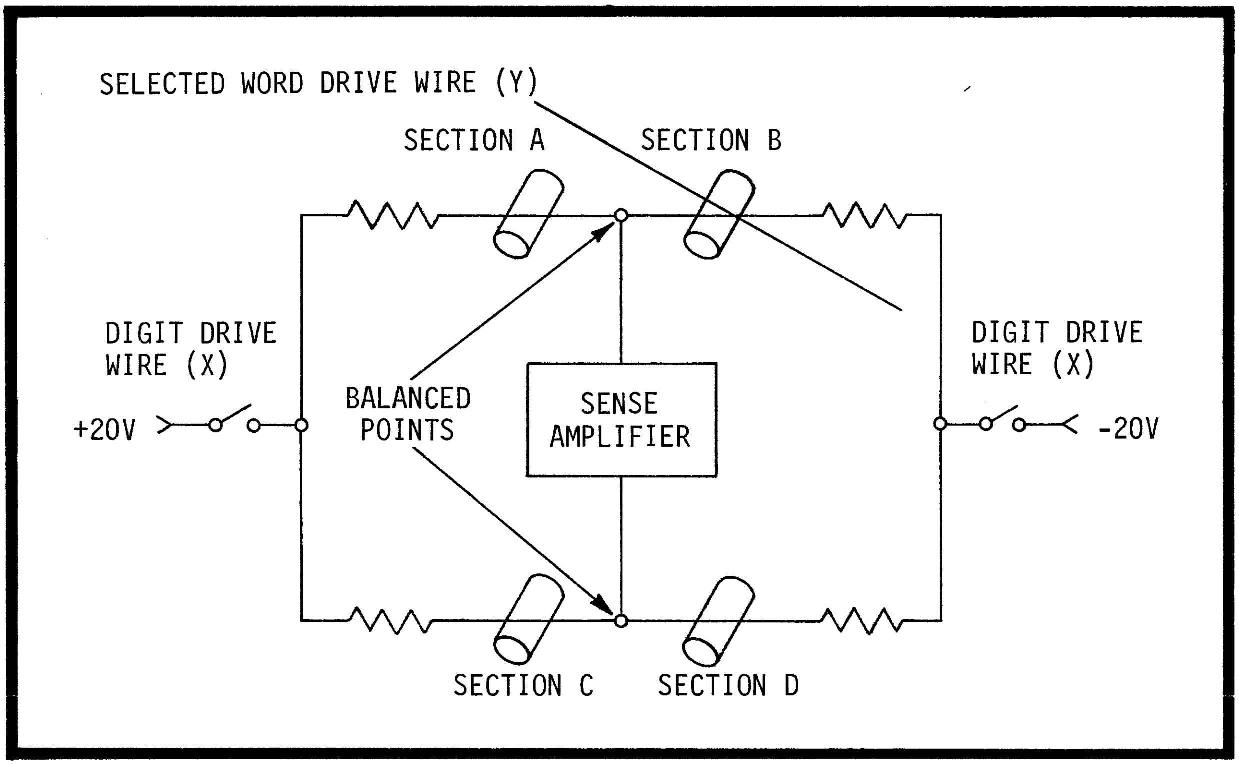
C. The rod is said to be half-selected since current flows only in the X wire. The rod retains its (B) polarity, because its characteristics are such that only coincident current is sufficient to induce a change in magnetic state.

D. The rod receives cancelling currents from the X and Y windings. The algebraic sum of these currents is 0, so the rod remains unchanged magnetically.

During a read operation, if a rod is flipped, then it is said to contain a 1-bit. If the rod is not flipped then it contains a 0-bit. The state of the rod is established by a write operation.

## Principles of Reading and Writing

The illustration below is a simplified schematic of physical memory and should be used as a point of reference during the explanations that follow.



- Read

  Assume that digit drive (winding X) current is applied by closing the switches. Sections A, B, C, and D of the diagram possess equal impedance if the states of the rods are not considered. The inputs to the sense amplifier are said to be balanced, since they are at equal voltage. Assume that the rod in section B is the selected rod because of coincident current between the word drive (winding Y) and the digit drive. Memory circuitry is such that only one rod per sense amplifier can be selected at a time. If the rod in section B is in a 1-bit state, its field is magnetically opposite to the state induced by the coincident read current. As the read current nears maximum, the selected rod is flipped to the 0 state. This change in state induces a voltage that aids the applied voltage. This in turn means that the upper input to the sense amplifier receives a voltage pulse that is more positive than the lower input. In effect, the amplifier compares voltages and detects the imbalance as a 1-bit.

If the selected rod is in the 0-bit state, its field is magnetically the same as that induced by the coincident current. It is not flipped; the amplifier inputs remain in balance, and the bit is detected as a 0.

- Write

When a character is to be written into memory, it is logically examined for 0-bits and 1-bits. Those bit positions within the character that contain a 0-bit inhibit coincident current for their respective memory positions; those positions containing a 1-bit permit selection for their respective positions. Since all affected positions are cleared to 0-bits during the first part of a write operation, and since write current flow is opposite to read current flow, any bit that is selected during the second part of a write operation is flipped to the 1-bit state. Since 0-bits inhibit selection by cancelling currents, they remain in the 0 state.

## DATA REPRESENTATION

The eight bits in a byte may be used to represent two four-bit binary numbers (position values of 8, 4, 2, and 1), an eight-bit binary number (position values of 128, 64, 32, 16, 8, 4, 2, and 1) or an eight-bit NCR Century character (four zone bits and four digit bits):

## NCR CENTURY CODE CHART

| B4–B1 → / B8–B5 ↓ | | 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 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0000 | 0 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS | HT | 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 |

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

If the binary configuration above is considered as two 4-bit binary numbers, decimal (position) values are 5 and 6.

| 8 | 4 | 2 | 1 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

5 6

If the configuration is considered as a single 8-bit binary number, decimal (position) value is 86.

| 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

86

If the configuration is treated as a character in the NCR Century code, it is equivalent to "V". Note that in the code chart, b8 is always 0, limiting the number of possible characters to 128. This configuration conforms to the American Standard Code for Information Interchange (ASCII).

| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

zone bits digit bits

Numerical data can be represented in either binary or binary coded decimal (BCD) format. In binary form, all eight bits of a character are significant and arithmetic operations are performed on multiples of eight bits. The value of a single character ranges from 0 (all bits = 0) to 255 (all bits = 1). In BCD format, only the least significant bits (b1 - b4) of a character are considered in arithmetic operations. The result of an arithmetic operation on BCD numbers, however, contains the proper zone bits for the 8-bit representations shown on the code chart. BCD data may be packed, with two 4-bit digits per byte; however, the processor must unpack the data before performing arithmetic operations.

## Binary Arithmetic

Digital computers use the binary system as the basis for all arithmetic operation since computer memory cores can only assume one of two positions and the binary system uses only two digits, 0 and 1.

- **Binary Addition**

There are four basic rules for binary addition:

0 + 0 = 0

0 + 1 = 1

1 + 0 = 1

1 + 1 = 10

Following these four rules, binary numbers are added directly as follows:

$$\begin{array}{l} 00111101 \\ \underline{00010110} \\ 01010011 \end{array}$$

The computer, however, cannot add these two numbers directly. A problem arises with respect to carries (i.e., 1 + 1 actually equals 0 with a carry of 1 to the next digit position, just as in decimal arithmetic, 9 + 1 equals 0 with a carry of 1 to the next decimal position). Instead of adding any carries at the time the two numbers are added, the computer adds the two numbers regardless of carries to obtain a partial sum, and then adds the carries to the partial sum at the proper digit locations to arrive at a final sum:

$$\begin{array}{ll} 00111101 & \\ \underline{00010110} & \\ 00101011 & \text{(Partial sum)} \\ \underline{\phantom{00}1\ 1\phantom{0000}} & \text{(Carries)} \\ 01010011 & \text{(Final sum)} \end{array}$$

- **Binary Subtraction**

The computer also has the capability of performing subtraction. The four rules for binary subtraction are:

0 - 0 = 0

1 - 1 = 0

1 - 0 = 1

0 - 1 = 1 with a borrow from the next 1-bit to the left (when 1 is borrowed from a digit, that digit is reduced to zero, and all intervening 0 digits that are passed over are reset to 1).

Following these rules, a direct binary subtraction takes place as shown:

$$\begin{array}{r} 00111101 \\ \underline{00010110} \\ \hline 00100111 \end{array}$$

Subtraction can also be performed by complementary addition. Thus, in the decimal system, the subtraction 7 - 4 can be performed in one of two ways:

| Direct Subtraction | Subtraction by Complementary Addition |
|---|---|
| $\begin{array}{r} 7 \\ -4 \\ \hline 3 \end{array}$ | $\begin{array}{r} 7 \\ +6 \\ \hline 13 \end{array}$ (6 is the 10's complement of 4; the 10's complement of a number is the difference between that number and 10) |
| Carry is ignored to compensate for using 10's complement. | |

The 10's complement is used for subtraction by complementary addition in the decimal system, since the base for this system is 10. For subtraction by complementary addition in the binary system, the 2's complement of a number is used, since the base for the binary system is 2. The 2's complement for a binary number is formed by subtracting that number from the next higher power of 2. Thus, the 2's complement of 00010110 is 11101010:

$$\begin{array}{rrl} 0 & 111111\phantom{00} & \text{(See rule for 0 - 1, above)} \\ \cancel{1} & \cancel{0}\cancel{0}\cancel{0}\cancel{0}\cancel{0}\cancel{0}00 & \text{- Next higher power of 2} \\ & 00010110 & \\ \hline & 11101010 & \end{array}$$

The 2's complement of a binary number may also be formed by first replacing each 0-bit with a 1-bit and each 1-bit with a 0-bit to form the 1's complement. If a 1 is then added to the rightmost bit of the 1's complement, the result is the 2's complement of the number:

| Binary Number | 1's Complement | |
|---|---|---|
| 00010110 | $\begin{array}{r} 11101001 \\ 1 \\ \hline 11101010 \end{array}$ | 2's complement |

The direct binary subtraction illustrated previously, could be performed by 2's complement addition as follows:

```
     00111101
     11101010   (2's complement of 00010110)
     --------
1    00100111
```

The carry is ignored to compensate for using the 2's complement.

Direct subtraction is not desirable for computer operations. Instead, the computer uses a variant of 2's complement addition to perform binary subtraction:

1. As the subtrahend is input to the adder (see ARITHMETIC LOGIC UNIT), its 1's complement is formed by the logic circuitry. The 1's complement of a number is derived by setting all 0-bits to 1 and all 1-bits to 0.

2. The subtrahend is added to the minuend, and carries are ignored, as explained in Binary Addition, to arrive at a partial sum.

3. The carries resulting from addition are added to the partial sum.

4. Also added at this time is an initial carry (1 is added to the least-significant bit of the number). Adding the initial carry at this point produces the same result as using the 2's complement would have produced, but it is more efficient for the computer to perform the operation this way than to have derived the 2's complement when the subtrahend was put into the adder.

```
   00111101
   11101001   (1's complement
   --------    of 00010110)

   11010100   (partial sum)
    1 1  1    (carries resulting
               from addition)
          1   (initial carry
               to derive 2's
               complement)
   --------
1  00100111   (Result)
```

By examining the final carry in a binary subtraction, the processor can determine whether the result of the subtraction was a positive or negative number. If there is a final carry, then the minuend was larger than or equal to the subtrahend and the result is a positive number (or zero). If there is no final carry, then the subtrahend was larger than the minuend, and the result is a negative number. The processor compensates for this condition if the result is to be used in subsequent arithmetic operations.

## BCD Arithmetic

The processor also has the capability of adding and subtracting decimal numbers coded in a binary format. In decimal addition, a correction factor (the 4-bit binary configuration for 6) is added to each digit of the A operand. The sum of this adder cycle is in turn added to the B operand digits to arrive at the result (always following the rules for binary addition):

```
      0100  (BCD 4)  First input - A operand digit
      0110  (Binary 6)
      ----
      0010  (Partial sum)
      1     (Carry)
      ----
      1010

      0111  (BCD 7)  Second input - B operand digit
      1010
      ----
      1101  (Partial sum)
       1    (Carry)
      ----
0001  0001  Final result (BCD 11)
```

In the illustration, the binary 6 was required, since the addition of 4 and 7 will produce an illegal BCD number (1011) without it. Since the 6 is always added during the first cycle, prior to the actual addition, it may at times have to be taken back from the result. For example, adding BCD 4 (0100) to BCD 3 (0011) produces BCD 7 (0111). However, since binary 6 is added to 4 before the addition, an illegal BCD number results (1011); therefore, the binary 6 must be extracted from the result in this case. The adder has the necessary circuitry to correct the result, and the whole operation would take place as follows:

```
      0100  (BCD 4)  First input - A operand digit
      0110  (Binary 6)
      ----
      0010
      1     (Carry)
      ----
   ┌──1010
   │
   │  0011  (BCD 3)  Second input - B operand digit
   └──1010
      ----
      1001  (Partial sum)
       1    (Carry)
      ----
      1101  (Uncorrected sum)
      0110  (Binary 6 correction)
      ----
      0111  (Corrected sum - BCD 7)
```

The processor determines whether the binary 6 correction is required by checking the carry beyond the most significant bit of the BCD number. If no carry is generated, or if the number is illegal, correction is required; if a carry is generated by the addition, and the number is legal (Binary Value ≤ 9), then correction is not required.

- BCD Subtraction

  The processor can also perform BCD subtraction by complementary addition. The only difference between BCD and binary subtraction by the processor is that the result of the BCD complementary addition may need binary 6 correction, just as the BCD addition did. The sequence of BCD subtraction by complementary addition is as follows:

1. Derive the 1's complement of the subtrahend.

2. Subtract the subtrahend from the minuend, using complementary addition to obtain the partial result.

3. Add initial carry and generated carries to obtain the uncorrected result.

4. Correct with binary 6 if required.

The rules governing these operations are as follows:

1. If there is a carry beyond the most significant bit of each digit after all carries have been added to the partial sum, a legitimate subtract operation has occurred (minuend was equal to, or larger than, subtrahend). The binary 6 correction is not required.

2. If there is no carry beyond the most significant bit of each digit after all carries have been added to the partial sum, the result is not legitimate, and the binary 6 correction must be made.

```
Example 1      5 - 4 = 1

               0101  (BCD 5)
               1011  (1's complement of BCD 4)
               1110  (Partial sum)
                 11  (Carries)
           1   0001  (BCD 1 - final carry is ignored when storing sum, but used
                      by processor to determine whether result is valid.  In this
                      case, the presence of the carry indicates that the result is
                      valid and does not require binary 6 correction).
Example 2      4 - 5 = -1

               0100  (BCD 4)
               1010  (1's complement of BCD 5)
               1110  (Partial sum)
                  1  (Carry)
               1111  (Uncorrected sum.  No carry beyond most significant
                      digit indicates to processor that result is negative.)
               0110  (Binary 6 correction - in effect, binary 6 is subtracted)
               1001  (Corrected sum. Negative results from BCD subtraction
                      are stored as the 10's complement of the absolute
                      value of the negative number.  Thus:  10's complement
                      of 1 is 9 (1001).  This procedure simplifies use of
                      these results in subsequent operations).
```

## ADDRESSING

Character locations in memory are numbered consecutively, starting at 0; each number is the address of one byte. A group of contiguous bytes is called a field and is addressed by the leftmost byte (most significant character); the length of a field can vary from 1 to 256 bytes. In the example below, the five-character field, A, containing the data 42385, is referenced by the address 100. If the same data were divided into two fields, B and C, the address of field B, containing the data 423, would be 100, and the address of field C, containing the data 85, would be 103.

FIELD A

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

FIELD B FIELD C

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

For purposes of simplification, the addresses in the example are given in decimal form. Actual memory addresses are in binary form, consisting of 16 bits (two bytes). The sixteen bits have a possible maximum value of 65,535; however, attempting to access a location equal to or greater than memory size causes a program error condition (PE). Consequently, b16 and b15 must be 0 when addressing a 16,384 byte memory, and b16 must be 0 when addressing a 32,768 byte memory.

To simplify address entry through the operator's console, the hexadecimal notation system is used. Entering the address 13,558 through the console in binary form, for example, would require the operator to set 16 switches to the configuration 0011 0100 1111 0110. Entering the same address in hexadecimal notation requires setting only four switches to the configuration 34F6. The conversion between hexadecimal and binary is quite simple; hexadecimal numbers are expressed to the base 16 and related to binary and decimal numbers as illustrated in the following table:

**DECIMAL - BINARY - HEXADECIMAL CONVERSION**

| DECIMAL | BINARY | HEXADECIMAL |
|---|---|---|
| 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 convert from binary to hexadecimal, separate the binary address into four 4-bit groups and assign the appropriate hexadecimal digit to each group:

$$0011010111101010\ \underbrace{0011}_{3}\quad\underbrace{0101}_{5}\quad\underbrace{1110}_{E}\quad\underbrace{1010}_{A} \quad = \quad 35EA$$

To convert from hexadecimal to binary, simply reverse the procedure:

$$2F9A \quad = \quad \underbrace{2}_{0010}\quad\underbrace{F}_{1111}\quad\underbrace{9}_{1001}\quad\underbrace{A}_{1010} \quad = \quad 0010111110011010$$

Should it become necessary to convert from hexadecimal to decimal, each position of the hexadecimal number must be expanded as follows:

$$\begin{aligned}34F6 &= (3 \times 16^3) + (4 \times 16^2) + (15 \times 16^1) + (6 \times 16^0)\\ &= (3 \times 4{,}096) + (4 \times 256) + (15 \times 16) + (6 \times 1)\\ &= 12{,}288 + 1024 + 240 + 6\\ &= 13{,}558\end{aligned}$$

Hexadecimal/decimal conversion can be simplified by using the following table:

HEXADECIMAL/DECIMAL CONVERSION TABLE

| Hexadecimal Equivalent | Position 4 | Position 3 | Position 2 | Position 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 a hexadecimal address to the equivalent decimal address, simply select the decimal value from each of the four columns that corresponds to the hexadecimal digit and add the values:

$$\begin{aligned}3FOB &= \quad 3 \qquad\quad F \qquad\quad 0 \qquad\quad B\\ &= 12{,}288 + 3{,}840 + 0 + 11\\ &= 16{,}139\end{aligned}$$

To convert from a decimal address to a hexadecimal address, a series of subtractions is used:

Decimal Address = 16,139

1. Locate the decimal value in the most significant position (position 4) that is equal to, or <u>less than</u>, the decimal address. Subtract this value from the decimal address.

```
   16139
 - 12288    = 3 (Position 4)
   -----
    3851
```

2. From the Position 3 column, select the decimal value that is equal to, or less than, the difference from the subtraction and subtract this number from the difference.

```
    3851
 -  3840    = F (Position 3)
    ----
      11
```

3. Repeat Step 2 for the two remaining positions.

```
      11
 -     0    = 0 (Position 2)
      --
      11
 -    11    = B (Position 1)
      --
       0
```

Hexadecimal Address = 3FOB

## INDEX REGISTERS

The NCR Century 100 memory contains 63 index registers, designed IR1 through IR63. Each register is the rightmost two bytes of a four-byte index register word; words are located consecutively in memory locations 0004 through OOFF (4 through 255) as illustrated on the next page.

## INDEX REGISTERS AND RESERVED MEMORY AREAS

| | | | Decimal Address | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | Hex Address |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SUPERVISOR CONTROL REGISTERS | | | 00000 | $T_o$ | $T_o$ | $T_o$ | $T_o$ | $T_o$ | $T_o$ | $T_o$ | $T_o$ | 0000 |
| | | | 00001 | T | T | T | T | T | T | T | T | 0001 |
| | | | 00002 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0002 |
| | | | 00003 | $Q_1$ | $Q_2$ | $Q_2$ | $Q_2$ | $Q_2$ | $Q_2$ | $Q_2$ | $Q_2$ | 0003 |
| | Index Register Word 1 | | 00004 | | | OI | RI | | G | E | L | 0004 |
| | | | 00005 | NOT USED BY NCR CENTURY 100 | | | | | | | | 0005 |
| | | IR1 | 00006 | $A_2$ | $A_2$ | $A_2$ | $A_2$ | $A_2$ | $A_2$ | $A_2$ | $A_2$ | 0006 |
| | | | 00007 | $A_1$ | $A_1$ | $A_1$ | $A_1$ | $A_1$ | $A_1$ | $A_1$ | $A_1$ | 0007 |
| | Index Register Word 2 | | 00008 | R | R | R | R | R | R | $R_m$ | $R_m$ | 0008 |
| | | | 00009 | NOT USED BY NCR CENTURY 100 | | | | | | | | 0009 |
| | | IR2 | 00010 | $B_2$ | $B_2$ | $B_2$ | $B_2$ | $B_2$ | $B_2$ | $B_2$ | $B_2$ | 000A |
| | | | 00011 | $B_1$ | $B_1$ | $B_1$ | $B_1$ | $B_1$ | $B_1$ | $B_1$ | $B_1$ | 000B |
| | Index Register Word 3 | | 00012 | NOT USED BY NCR CENTURY 100 | | | | | | | | 000C |
| | | | 00013 | NOT USED BY NCR CENTURY 100 | | | | | | | | 000D |
| | | IR3 | 00014 | S | S | S | S | S | S | S | S | 000E |
| | | | 00015 | S | S | S | S | S | S | S | S | 000F |
| USER CONTROL REGISTERS | Index Register Word 4 | | 00016 | $T_o$ | $T_o$ | $T_o$ | $T_o$ | $T_o$ | $T_o$ | $T_o$ | $T_o$ | 0010 |
| | | | 00017 | T | T | T | T | T | T | T | T | 0011 |
| | | IR4 | 00018 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0012 |
| | | | 00019 | $Q_1$ | $Q_2$ | $Q_2$ | $Q_2$ | $Q_2$ | $Q_2$ | $Q_2$ | $Q_2$ | 0013 |
| | Index Register Word 5 | | 00020 | | | OI | RI | | G | E | L | 0014 |
| | | | 00021 | NOT USED BY NCR CENTURY 100 | | | | | | | | 0015 |
| | | IR5 | 00022 | $A_2$ | $A_2$ | $A_2$ | $A_2$ | $A_2$ | $A_2$ | $A_2$ | $A_2$ | 0016 |
| | | | 00023 | $A_1$ | $A_1$ | $A_1$ | $A_1$ | $A_1$ | $A_1$ | $A_1$ | $A_1$ | 0017 |
| | Index Register Word 6 | | 00024 | R | R | R | R | R | R | $R_m$ | $R_m$ | 0018 |
| | | | 00025 | NOT USED BY NCR CENTURY 100 | | | | | | | | 0019 |
| | | IR6 | 00026 | $B_2$ | $B_2$ | $B_2$ | $B_2$ | $B_2$ | $B_2$ | $B_2$ | $B_2$ | 001A |
| | | | 00027 | $B_1$ | $B_1$ | $B_1$ | $B_1$ | $B_1$ | $B_1$ | $B_1$ | $B_1$ | 001B |
| | Index Register Word 7 | | 00028 | NOT USED BY NCR CENTURY 100 | | | | | | | | 001C |
| | | | 00029 | NOT USED BY NCR CENTURY 100 | | | | | | | | 001D |
| | | IR7 | 00030 | S | S | S | S | S | S | S | S | 001E |
| | | | 00031 | S | S | S | S | S | S | S | S | 001F |
| WORK REGS | Index Register Words 8-32 | | 00032 | RC | RC | RC | RC | RC | RC | RC | RC | 0020 |
| | | | 00033 | X | X | X | X | X | X | X | X | 0021 |
| | IR Words 33 - 63 | | 00127 | X | X | X | X | X | X | X | X | 007F |
| | | | 00128 | PC | PC | PC | PC | PC | PC | PC | PC | 0080 |
| | | | 00129 | X | X | X | X | X | X | X | X | 0081 |
| | | | 00255 | X | X | X | X | X | X | X | X | 00FF |

Key (See COMMANDS Section)

$T_o$ = T Tally Register

T = Original T Value (Field Length)

$Q_1$ = Command Size (0 = 2-address)
(1 = 1-address)

$Q_2$ = Command Code

OI = Overflow Indicator
RI = Repeat Indicator
G = "Greater" Flag
E = "Equal" Flag
L = "Less" Flag

$A_2A_1$ = Effective A Address

R = Last R Character ($R_A$ or $R_B$)
$R_m$ = Mode of Addressing 00 - no Incremental Indexing
11 = Incremental Indexing

$B_2B_1$ = Effective B Address

S = Sequence Control Register

RC = Repeat Counter

PC = Temporary Storage Used by Printer Cntrl
X = Miscellaneous Index Register Words

Note that Supervisor and User Control Registers are completely symmetrical, obviating the necessity of save and restore operations when the processor enters a trap routine.

To maintain compatibility with larger NCR Century Systems, NCR Century 100 programs must not reference locations 0 through 31 (0000 - 001F).

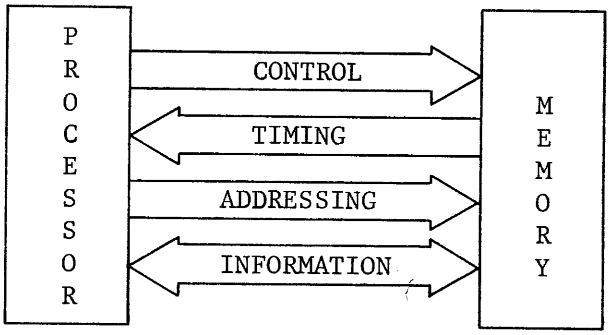
Locations 256-1279 (0100-04FF) are reserved for I/O control and resident executive use.

The use of index registers in NCR Century Series computers provides the programmer considerable flexibility in manipulating data fields. As discussed in the Commands section of this publication, index registers in effect permit three separate modes of memory addressing on the NCR Century 100, with additional modes available on larger NCR Century systems. Note that IR1 through IR7 are reserved for hardware and software operations within the processor. There is no hardware protection provided for index registers. This means that the programmer must never access these locations indiscriminately, since their contents can be altered.

## FUNCTIONAL DESCRIPTION

### Introduction

The function of the NCR Century memory unit is to receive information from the processor (ALU or I/O Control), retain it, and, upon request, return it to the processor. Communication between processor and memory falls into four general categories: control, timing, addressing, and information transfer.



- Control

  Control from the processor performs two basic functions: initiation of the timing sequence, and request of a read or write operation.

- Timing

  Timing from memory to the processor is based upon three hardware-generated signals. These signals occur at specific times during the memory cycle and control processor timing (one memory cycle requires 800 nanoseconds).

- Addressing

  Addressing from the processor to memory takes place over 14 communication lines (15 lines if 32K memory option included). The address on these lines is decoded by memory hardware to provide character (byte) addressability.

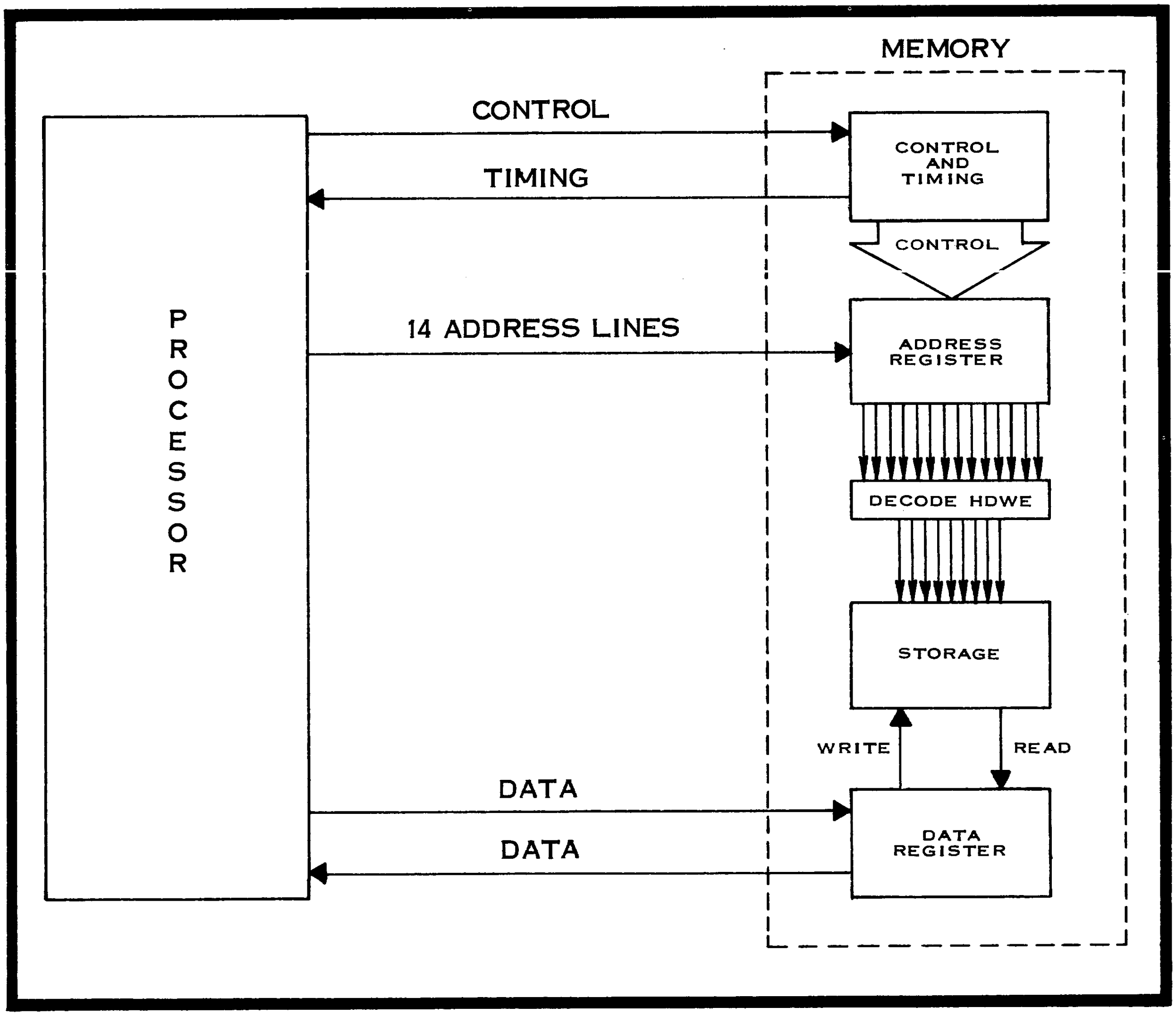
- Information Transfer

  Information is transferred between the processor and memory one byte at a time, including the parity bit. This information may be either commands or data.

## Functional Operation

NCR Century 100 memory performs two basic operations, a read-restore operation and a clear-write operation, each requiring one memory cycle. The processor informs memory during one cycle that a read or write operation will be performed during the following cycle. At the beginning of the cycle, the memory address register accepts an address from the processor, decodes it, and selects the proper location in the memory storage area. If the operation is a read-restore, during the first part of the cycle, 1-bits detected in the selected area are transferred to the data register where the byte is assembled and transmitted to the processor. During the second half of the operation, the information that was read from memory is restored to the same location that it was read from, completing the cycle.

A write operation also oonsists of two functions performed during a single memory cycle. Once the area has been selected, it is cleared so that all bits are 0. During this time, the processor has transmitted the data to be written into the data register. During the second half of the cycle, the data is written into the selected area by setting the appropriate bits to 1.



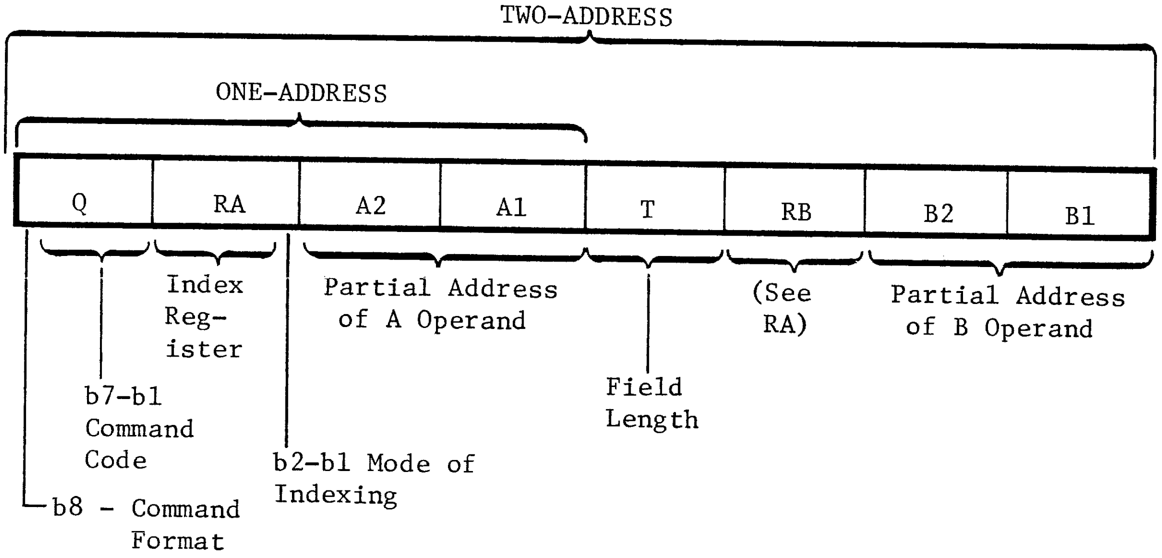
# ARITHMETIC LOGIC UNIT

## INTRODUCTION

The functional description of NCR Century ALU operation contained in this section is hardware-oriented and must not be construed as a programming guide. All definitions and explanations assume that a program written in NEAT/3 language has been compiled and is resident in memory in a form that can be used by the processor hardware.

## COMMANDS

Object-program commands are stored in memory in one- or two-address formats, occupying four or eight bytes. Even though most commands require two operand addresses (an add command, for example, obviously requires the addresses of the two operands to be added), the two formats are functionally equivalent since the one-address form uses the T and B addresses from the preceding command. In general, the results of an arithmetic operation replace the contents of the original B operand address.



### Command Code - Q

The Q portion of the command specifies the command code and the command format. The binary value of b7 - b1 designates the command to be executed (add, subtract, etc.). The most significant bit (b8) indicates the command format:

b8 = 0 The command is two-address format.
b8 = 1 The command is one-address format; the addresses of the T and B characters must be obtained from the preceding command.

## Index Register - RA

The RA portion of the command specifies whether indexing is to be performed, the index register to be used, and which of two modes of indexing is required:

1. The binary value of b8 - b3 indicates the location of one of the 63 index registers. If this value is 0, no indexing is required.

2. If indexing is specified (b8 - b3 ≠ 0), the two least significant bits (b2 - b1) indicate the mode of indexing:

   b2b1 = 0  Incremental indexing mode is not required.
   b2b1 = 3  Incremental indexing mode is required.

3. If either mode of indexing is specified, the contents of the index register are added to the contents of the partial A address (A2A1) to derive the address of the A operand:

   Assume that the contents of RA = 00034 (decimal), the address of IR8, and that IR8 contains the number 02000.

   If the contents of A2A1 are 00315, then:

   02000 + 00315 = 02315 (address of the effective A operand)

   If b2b1 of RA = 0, the index register would not be affected by the addition, but would continue to contain "02000" following the operation.

   If b2b1 of RA = 3, incremental indexing would be specified. That is, following the addition, "02315" would replace "02000" as the contents of IR8.

   Incremental indexing means that a program can "step through" a memory area - a block of fixed-length records or a table, for example - deriving the next significant address automatically each time the operation is performed.

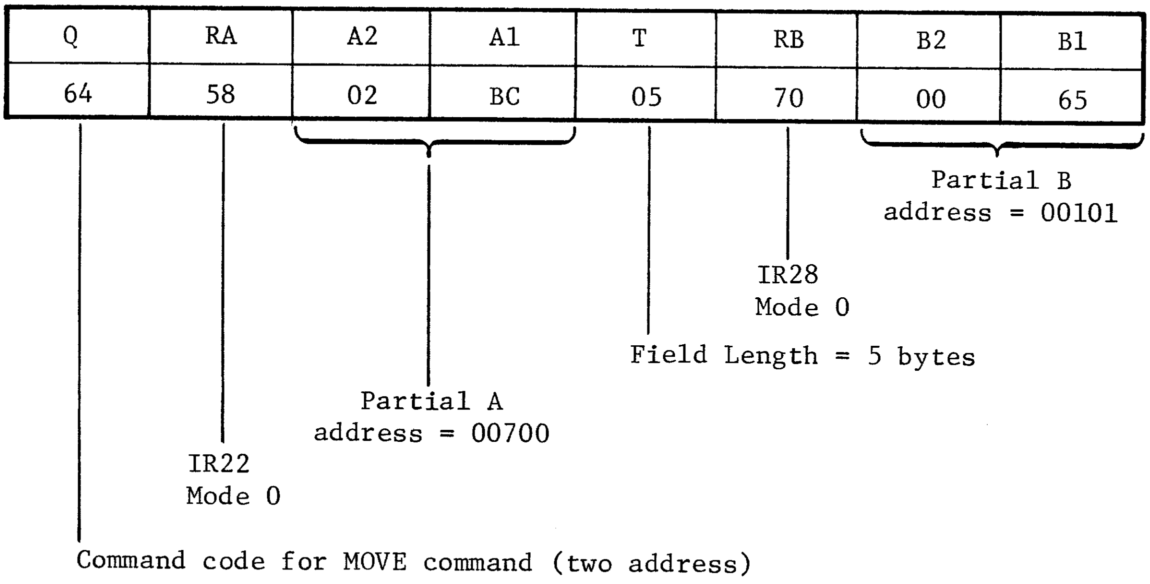
   Obviously, if RA = 0, A2A1 is the effective A operand address.

   The effective B operand address is derived in the same way, using RB and B2B1.

## Length - T

The binary value of the T portion of the command specifies the field length in bytes of the A and B operands. T is an 8-bit field ranging in value from 0 to 255 with 0 equal to 256. The lengths of the A and B operands are the same except for PACK and UNPACK commands. For a PACK command, T specifies the length of the A operand and T/2 (if T is even) or (T + 1)/2 (if T is odd) specifies the length of the B operand. For an UNPACK command, T is the length of the A operand and 2T is the length of the B operand.

Command Example (Values in hexidecimal notation)



| Q | RA | A2 | A1 | T | RB | B2 | B1 |
|---|---|---|---|---|---|---|---|
| 64 | 58 | 02 | BC | 05 | 70 | 00 | 65 |

Partial B address = 00101
IR28 Mode 0
Field Length = 5 bytes
Partial A address = 00700
IR22 Mode 0
Command code for MOVE command (two address)

Assume that the two index registers contain the following information:

(IR22) = OAFO (02800)
(IR28) = 0898 (02200)

OAFO + 02BC = ODAC (03500) Effective A address
0898 + 0065 = 08FD (02301) Effective B address

Operand contents before command execution:

A =

| 03500 | 03501 | 03502 | 03503 | 03504 |
|---|---|---|---|---|
| 00110100 | 00110001 | 00110010 | 00110110 | 00111001 |

B =

| 02301 | 02302 | 02303 | 02304 | 02305 |
|---|---|---|---|---|
| 00110001 | 00110010 | 00110110 | 00111001 | 00110000 |

Following execution of the MOVE command, the initial contents of A have replaced the initial contents of B. The contents of A are unchanged:

A =

| 03500 | 03501 | 03502 | 03503 | 03504 |
|---|---|---|---|---|
| 00110100 | 00110001 | 00110010 | 00110110 | 00111001 |

B =

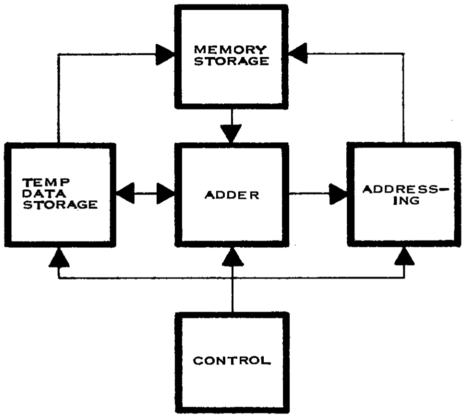
| 02301 | 02302 | 02303 | 02304 | 02305 |
|---|---|---|---|---|
| 00110100 | 00110001 | 00110010 | 00110110 | 00111001 |

Since the command specified no incremental indexing, the contents IR22 and IR28 remain the same. If the command had specified incremental indexing for both operands, then the contents of IR22 would be ODAC, and the contents of IR28 would be 08FD at the termination of the command.

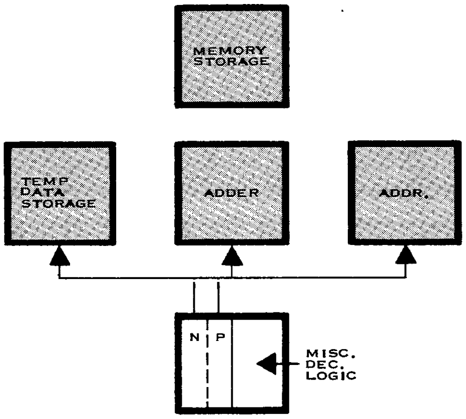
The contents of T are unchanged by the operation.

## FUNCTIONAL OPERATION

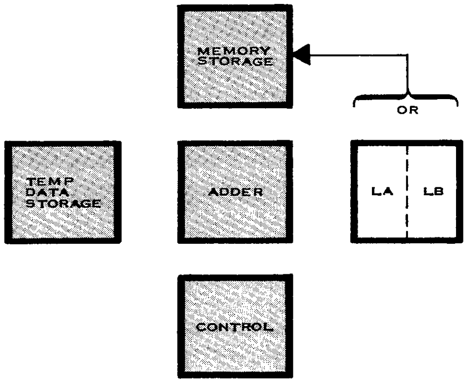
For purposes of discussion, the processor logic that performs the actual command execution (hardware) is simplified in the block diagram at the right. When the command is decoded, the control section is activated to regulate data transfer among the temporary data storage register, the adder, and the addressing logic.



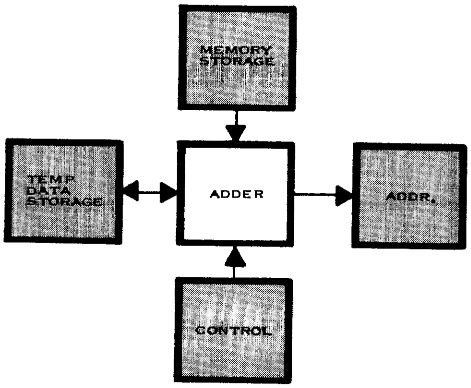
The control logic is divided into three sections, designated N, P, and Miscellaneous Decision Logic in the second diagram. The section designated N is a counter that controls the processor as it proceeds through the series of steps required to execute the command. These steps are called N flows and may be separated into three groups: the flows involved with setting up the command, those required to interpret the command, and those required to execute the command. Within each N flow there is a series of computer cycles called P counts that control timing for the N flow. The number of P counts within an N flow varies from 1 to 11, depending upon the function to be performed. The miscellaneous decision logic guides the processor through the necessary N flows to accomplish the desired result and controls the P counter and any other logic pertinent to the particular command.



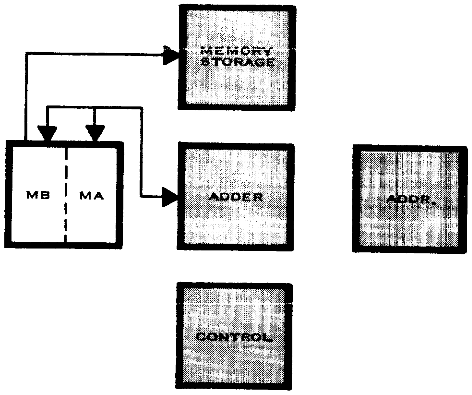
As the processor proceeds through the command flows, it is necessary to read from or write into memory during every cycle (P count). Before reading or writing, the proper memory address must be accessed. There are two types of addresses that may be transmitted to memory: one type accesses the special areas of memory reserved for the storage of control words, flags, control registers, etc. (See subsequent sections of this publication for a detailed explanation of these special areas.) These special locations are addressed by the logic labeled LB in the diagram. Index registers and all addresses called for by the program are addressed by the logic labeled LA. These locations can contain data relevant to a problem being solved, data to be output to a peripheral device, and so forth.



At the core of the NCR Century processor operations is a hardware area called the adder. The primary function of the adder is arithmetic. It is here that additions, subtractions, comparisons and the like are performed. The adder is also used as an intermediate stage for most data transfer within the processor.



There are two temporary data storage registers in the NCR Century 100 processor, labeled MA and MB in the diagram; each stores one byte of data. MA is used strictly for storage, while MB may also serve as an I/O buffer, output data to memory during a write operation, or accept data entered through the console input switches.

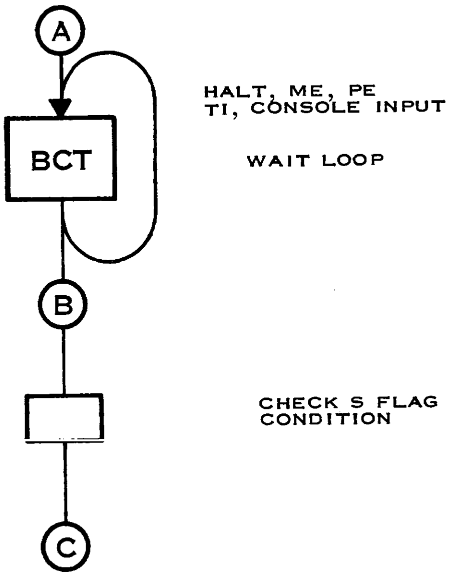


## Command Setup, Indexing, Interpretation, and Execution

NCR Century 100 hardware operates on a command in three distinct phases: setup (including indexing), interpretion, and execution. In the diagram accompanying this description, between-commands-testing (BCT) is indicated as the logical entry point to command flow for the sake of completeness. The BCT procedure is explained in detail in a subsequent section of this manual.

### • Command Setup and Indexing

In the command setup phase, the processor reads the command from memory and writes the command code, effective A and B addresses, and length character in the special index registers used for command setup.



Command setup begins with a test of the hardware S flag to determine the current state of the processor. The processor has two states of operation, user and supervisor. As the name implies, the processor is in the user state whenever user programs are operating normally; it switches to the supervisor state for I/O termination interrupts (see I/O Control section), errors, or commands not recognized as part of the legal set. In the supervisor state, control information is stored in memory locations 0 through 15; a symmetrical set of registers in locations 16 through 31 stores control information in the user state.

The processor turns the S flag ON when:

1. A command is issued that is not one of the hardware-recognized set.

2. The processor, in the user state, is ready to access the next command and an I/O termination condition exists (see I/O Control section). II

3. An ME or a PE condition is detected.

4. The LOAD button on the operator's console is pressed while the processor is in a Halt state.

The S flag setting determines which of the two sequence control registers (see memory map, page 19 ) governs the command setup; this register is stored in the memory address register.

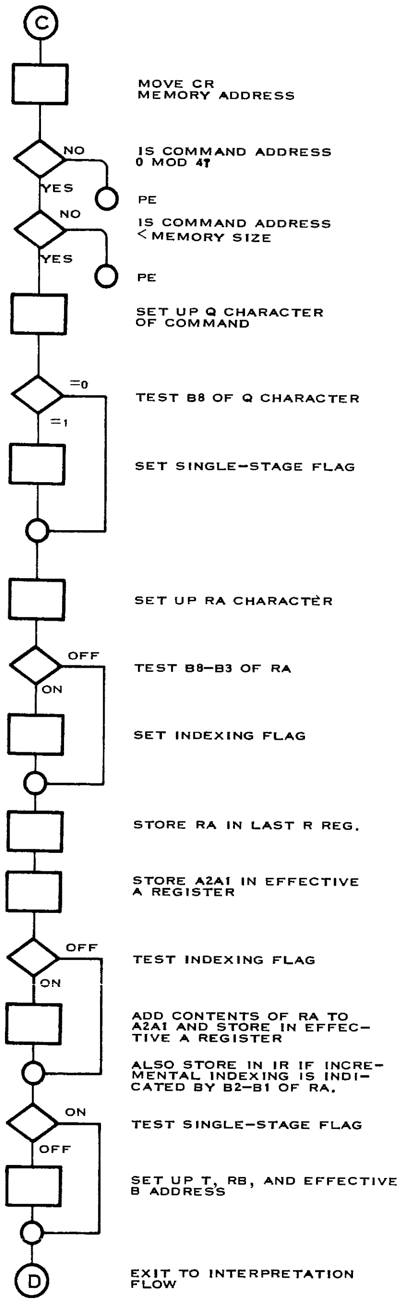
The command address is read from the sequence control register and tested. If this address is greater than memory size, or not evenly divisible by 4, the processor enters a program error (PE) routine.

If the command address is acceptable, the processor then reads the command Q character and tests b8 to determine whether the command is single- or double-stage. If the commmand is single-stage, the hardware single-stage flag is turned on (bit set to 1); if the command is double-stage, the flag bit is set to 0. During the next cycle (P count), the Q character is written into the appropriate index register location.

During the following P count, the RA character is read and bits 8 - 3 are tested. Any 1-bit detected in this area sets a hardware flag to indicate that indexing is required. The RA character is written in the "last R" register. During the next four cycles, the A2A1 character is read from memory and written into the effective A register, one byte at a time.

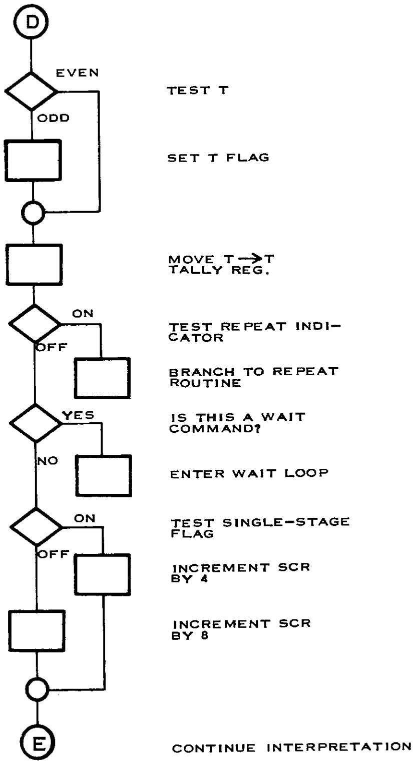
If the flag has been set to indicate indexing, the processor reads out the contents of the designated index register, adds these to the A2A1 character, and stores the result back into the effective A address register. If incremental indexing is required (b2b1 of RA on), the result is also stored in the designated index register.

The single-stage flag is tested. If it is on, the processor enters the command interpretation phase. If the flag is off, the T character (length) is stored in the appropriate register, and the RB, B2B1 portions of the command are set up in the same manner as the RA, A2A1 portions before entering the interpretation phase.


C
MOVE CR MEMORY ADDRESS
NO
IS COMMAND ADDRESS 0 MOD 4?
YES
PE
NO
IS COMMAND ADDRESS < MEMORY SIZE
YES
PE
SET UP Q CHARACTER OF COMMAND
=0
TEST B8 OF Q CHARACTER
=1
SET SINGLE-STAGE FLAG
SET UP RA CHARACTÈR
OFF
TEST B8-B3 OF RA
ON
SET INDEXING FLAG
STORE RA IN LAST R REG.
STORE A2A1 IN EFFECTIVE A REGISTER
OFF
TEST INDEXING FLAG
ON
ADD CONTENTS OF RA TO A2A1 AND STORE IN EFFECTIVE A REGISTER
ALSO STORE IN IR IF INCREMENTAL INDEXING IS INDICATED BY B2-B1 OF RA.
ON
TEST SINGLE-STAGE FLAG
OFF
SET UP T, RB, AND EFFECTIVE B ADDRESS
D
EXIT TO INTERPRETATION FLOW

- Command Interpretation

In the command interpretation phase, the contents of the T register (length) are stored in the T tally register; the command code and certain flags are read from memory and analyzed; certain branches are executed if required, and the processor is directed to the proper command flow.



The T character is read from memory and tested. If the character is odd, a hardware flag is set for possible use by a PACK command (if a PACK command is issued, the processor must know whether the affected length is odd or even, since the length of the B operand is computed as T/2 if T is even or as (T + 1)/2 if T is odd). For a double-stage command, the T value is the one stored during the command setup phase; for a single-stage command, the T value is one stored during a previous command setup.

During the next cycle (P count) the processor stores the T character in a tally register. The contents of the tally register will be decremented during the command execution phase as a control over operand addresses.

The processor then reads the Q portion of the command from memory and tests the condition of the flags at location 00004 (Supervisor State) or 00020 (User State).

1. If the repeat indicator is on (RI), the processor enters the repeating flow immediately, since the status of the indicator means that this command is being repeated a stipulated number of times.

2. If the command is WAIT, the processor enters a loop where it remains until the COMPUTE button on the console is pressed.

During the next cycle, the processor examines the single stage flag. If the flag is ON, the contents of the sequence control register are incremented by 4 so that the register will contain the address of the next command in sequence; if the flag is OFF, the contents of the register are incremented by 8.

If the command is one of the BRANCH commands, the processor examines the status of the G, L, and E flags and the Overflow Indicator (O/I). If the required condition has been met, the processor stores the address of the first command in the branch subroutine into the sequence control register.

If the command is not one of the hardware-recognized set, the S flag is turned on.

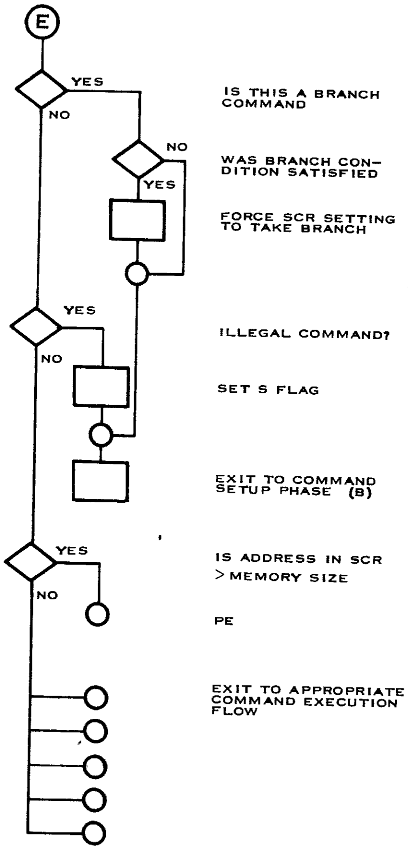
If the address in the control register is greater than memory size, the processor enters the error routine (PE).

Once the command has been interpreted, the processor enters one of six possible flows:

1. If the command is a BRANCH or an invalid command, the processor returns to the setup phase. If the S flag is on, the processor switches to the Supervisor State to handle the invalid command. If the S flag is off, the processor accesses the sequence control register to obtain the address of the next command. In this instance, this will be the address of the first command.in the branch subroutine.

2. The processor enters a common execution flow if the command is arithmetic (add, subtract, compare).

3. If the command is PACK, UNPACK, REPEAT, or requires an I/O operation, the processor enters one of the four special execution flows reserved for these commands.


E
YES
NO
IS THIS A BRANCH COMMAND
NO
YES
WAS BRANCH CON-DITION SATISFIED
FORCE SCR SETTING TO TAKE BRANCH
YES
NO
ILLEGAL COMMAND?
SET S FLAG
EXIT TO COMMAND SETUP PHASE (B)
YES
NO
IS ADDRESS IN SCR > MEMORY SIZE
PE
EXIT TO APPROPRIATE COMMAND EXECUTION FLOW

- Command Execution

The actual steps involved in executing a given command are subject to too many variables to permit detailed description within the scope of this publication. Once any command has been successfully executed, however, the processor returns to the setup phase and accesses the next command in sequence.

- Implied T and B Operation

All commands terminate execution with predictable addresses for T and B values available to the subsequent command. For all commands, the T value available is the one stored in the T register at the end of command setup; this value can be used through successive one-address commands. The B value is more variable and depends upon the specific command executed. For all the NCR Century 100 commands except PACK and UNPACK, however, the B value after execution is the same as the B value after command setup. Thus, any command that requires two operands may be coded in a one-address form, with the second operand and the length implied. The setup of a one-address command does not disturb the T and B values from the previous command, and these are used as if the current command had set them up. This characteristic permits strings of one-address commands to be "chained" to a two-address command as illustrated:

Given K, L, and M as memory locations, and the commands:

MOVE T characters from K → N
ADD L
ADD M
The equation K + L + M → N is executed as follows:

MOVE is a two address command that moves K to N and specifies the length and the B operand addresses for L and M. The contents of L and M are then added sequentially to the contents of N and stored there, the ADD L and ADD M commands using the implied T and B values from the MOVE operation.

## Miscellaneous Processor Operations

During processor operation, any or all of the following situations may occur:

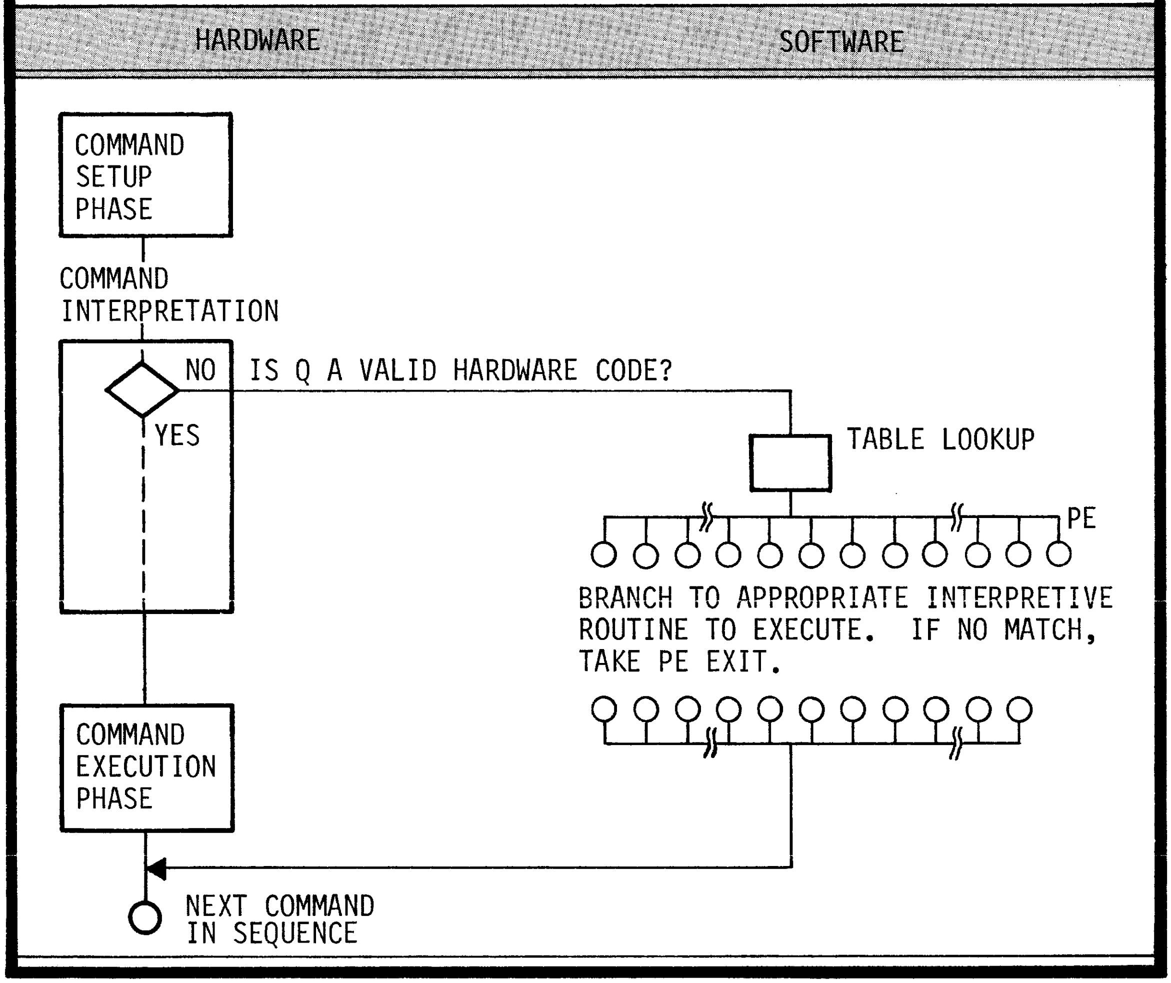
- Command code trapping
- I/O termination
- Memory error
- Program error

Century 100 hardware has built-in provisions for handling each situation.

- Command Code Trap

During the interpretation phase, the processor examines the command to determine whether it is one of the hardware-recognized set. If it is not, the processor sets the S flag ON and traps to the supervisor state. This causes the address of the next command to be accessed from the supervisor sequence control register, which, at this point, will contain the starting address of a software Executive routine. This routine examines the command code to determine whether it is actually invalid. If the command is invalid, the Executive directs the processor to enter a routine that halts the operation with an error display on the operator's console. It is possible, however, for the command to be valid even though not recognized by the hardware. In this case, the software Executive directs the flow to further subroutines where the command is interpreted and executed, first establishing the necessary linkages to effect a return to the user program once the command is executed.

COMMAND CODE TRAP ROUTINE



- I/O Termination

Processor operation for input/output termination is explained in the I/O Control section of this publication.

- Memory Error (ME)

Parity is checked for each character as it is read out of memory. Whenever a parity error is detected, control is transferred to the ME flow in BCT (see BETWEEN COMMANDS TESTING). Memory conditions are not changed.

The transfer of control occurs immediately upon discovery of the parity error except that the processor always completes the setup phase of the first four command characters.

If an ME is detected while accessing the control register, accessing any character during the execution phase, or accessing a character during an I/O control operation, the ME indicator is set on; the operation is terminated, and the BCT flow is entered.

An ME also turns on the error indicator (EI). If the EI is already on when the error occurs, the processor immediately enters the error halt state. To reset the error halt condition, the operator must set the HALT switch ON and activate the RESET switch.

- Program Error (PE)

  All data and command addresses are checked for validity. When any of the conditions listed below is discovered, control is transferred to the PE flow. Internal memory conditions are not changed.

  The recongnized program errors are:

  1. Any address greater than memory size.

  2. Any command location (control register address) that is not evenly divisible by 4.

  3. A printline address that is not evenly divisible by 4.

  4. Repeat indicator ON and the Repeat Counter equal to 0.

  Whenever a PE is detected, the PE indicator is set ON; the operation is terminated, and the BCT flow is entered. The PE also turns on the error indicator. If the EI is ON when the PE occurs, the processor immediately enters the error halt state. To reset the error halt condition, the operator must set the HALT switch ON and activate the RESET switch.

  If an ME or a PE is detected during the I/O Control portion of an operation, an error reset signal is sent to the selected peripherals on the active trunks and data transmission ceases. This action also inhibits the sending of an S3 status character (see I/O CONTROL section).

### Between Commands Testing (BCT)

Reference was made in the previous sections to between-commands-testing. BCT comprises all tests relating to errors or I/O conditions and results in either a halt for successive error conditions, or alteration of the supervisor control register contents to force entry to a new flow.

BCT is entered if an ME, PE, or peripheral termination occurs (see I/O CONTROL section). The ME and PE cause the command to terminate before it is completed. If a peripheral termination occurs, the current command is completed before BCT is entered.

BCT functional operation depends upon the conditions of the various flags and indicators involved:

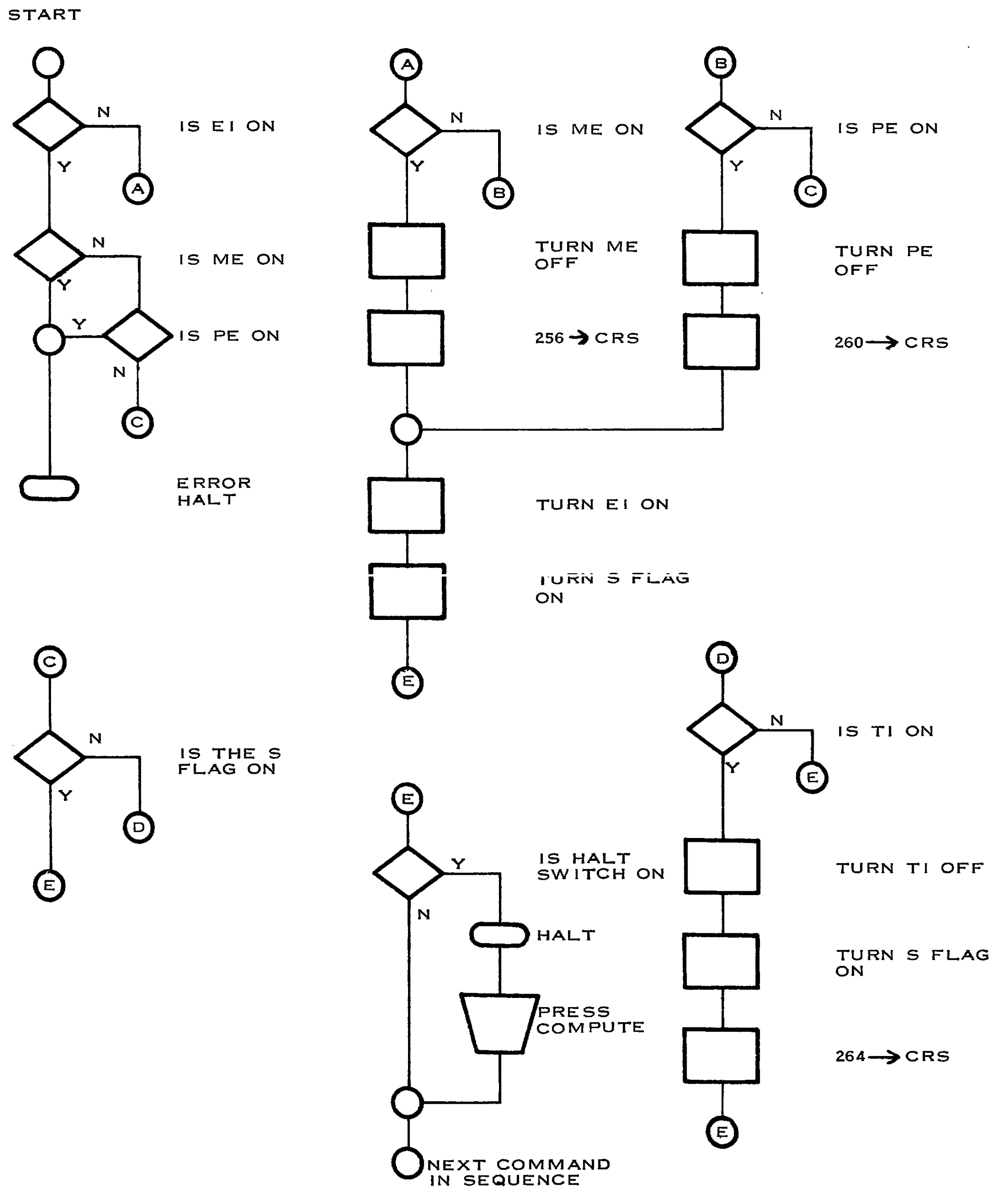
1. EI ON and either ME or PE ON: BCT causes the processor to enter a halt state. Active peripherals are not deselected, but all I/O processing requests from integrated and common trunk peripherals are ignored, except those from the integrated printer.

2. ME or ME and PE ON, EI OFF: BCT stores the address 00256 into the supervisor control register, sets ME and/or PE OFF, sets EI ON, sets the S flag ON. If HALT is ON, the halt state is entered; if HALT is off, the combination of the S flag and the address stored in the sequence control register sends the processor into the proper subroutine (00256 is the address of a BRANCH command to the ME trap routine). If COMPUTE is pressed while the processor is in a halt state, command processing is initiated.

3. PE ON, ME OFF, EI OFF: BCT stores the address 00260 into the supervisor control register, sets PE OFF, sets EI ON, and sets the S flag ON. If HALT is ON, the halt state is entered; if HALT is OFF, the combination of the S flag and the address stored in the sequence control register sends the processor into the proper subroutine (00260 is the address of a BRANCH command to the PE trap routine). If COMPUTE is pressed while the processor is in a halt state, command processing is initiated.

4. TI ON (no error): BCT puts the address 00264 in the supervisor sequence control register, sets TI OFF and the S flag ON. When the processor enters the command setup phase, the combination of the S flag and the sequence control register address send it to the I/O termination routine (see I/O CONTROL section).

## BCT FUNCTIONAL FLOW



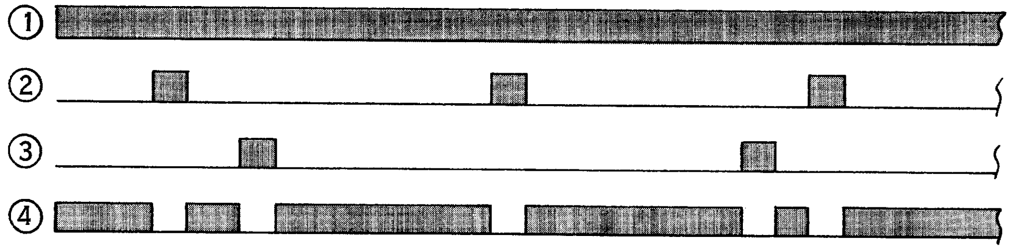
## INTRODUCTION

NCR Century I/O operation is initiated by the ALU, using information derived from an I/O command. After the ALU completes the selection sequence it returns to processing subsequent program instructions, leaving the I/O Control portion of the processor to direct data transfer. Should the next command in sequence be another INOUT command designating a peripheral attached to the second trunk, the procedure is repeated. Peripheral operation on either trunk must terminate before another peripheral can be selected on that trunk.

Data is input and output serially, by byte. The parity bit is checked by the I/O Control when data is received from the peripheral, and by the peripheral unit itself when it receives data from the processor. Six or eight memory cycles are normally required to transfer one byte, depending upon the peripheral and trunk.

NCR Century 100 logic is such that, if the I/O Control and the ALU request a memory cycle at the same time, the I/O control takes priority. Since peripheral servicing does not require all the memory cycles available within a given time frame, the ALU "steals" unused cycles and continues processing during data transfer. This characteristic provides the NCR Century 100 with effective three-way simultaneity; that is, it is possible for two I/O operations and an ALU operation to take place in a given space of time.

ILLUSTRATION OF 3—WAY SIMULTANEITY



① REPRESENTS THE CONTINUOUSLY AVAILABLE MEMORY TIME.

② REPRESENTS THE MEMORY TIME TAKEN BY A PERIPHERAL UNIT ON TRUNK 0 OPERATING CONTINUOUSLY AT FULL SPEED.

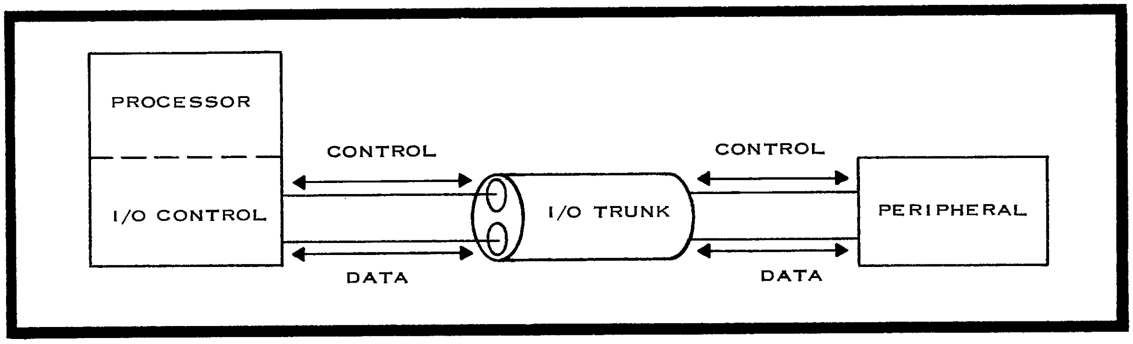
③ REPRESENTS THE MEMORY TIME TAKEN BY A PERIPHERAL UNIT ON TRUNK 1 OPERATING CONTINUOUSLY AT FULL SPEED.

④ REPRESENTS THE TIME AVAILABLE FOR INTERNAL PROCESSING WHILE TWO PERIPHERAL UNITS ARE OPERATING AT FULL SPEED.

## TRUNKS

The NCR Century 100 System has two input/output trunks, designated Trunk 0 and Trunk 1. Each trunk has eight positions to which a peripheral may be attached. NCR Century 100 logic gives Trunk 1 priority over Trunk 0.

Each trunk is actually a cable containing two separate groups of wires, as illustrated. The group labelled "Control Lines" is used to select the peripheral, initiate data transfer, and terminate the operation. There are 18 data lines, subdivided for input and output, one for each data bit and one for the parity bit.



The maximum number of bytes that a trunk can transfer per second is called the trunk bandwidth. Either trunk on the NCR Century 100 system has a theoretical bandwidth of 156 kb (156,000 bytes per second). In a system configuration, however, certain conventions must be adhered to for optimum I/O operation:

1. The NCR Century 100 cannot accommodate peripherals having transfer rates above 118 kb on either trunk.

2. No peripheral with a transfer rate exceeding 40 kb can operate concurrently with the integrated disc. Therefore, for effective I/O - compute simultaneity, Trunk 0 should be restricted to peripherals with a transfer rate of 40 kb or less.

3. The maximum total I/O rate that the NCR Century 100 can accommodate when both trunks are transferring data is 208 kb.

As a general rule, then, low-speed peripherals, such as punched tape and card readers or printers, should be assigned to Trunk 0, with Trunk 1 reserved for higher-speed magnetic file devices.

## PERIPHERAL TYPES

Two general classes of peripherals are used with the NCR Century Series: integrated devices that share some electronics and power supplies with the processor, and common trunk (freestanding) peripherals that are self-contained.

## Integrated Peripherals

NCR Century 100 integrated peripherals and their dedicated trunk positions are:

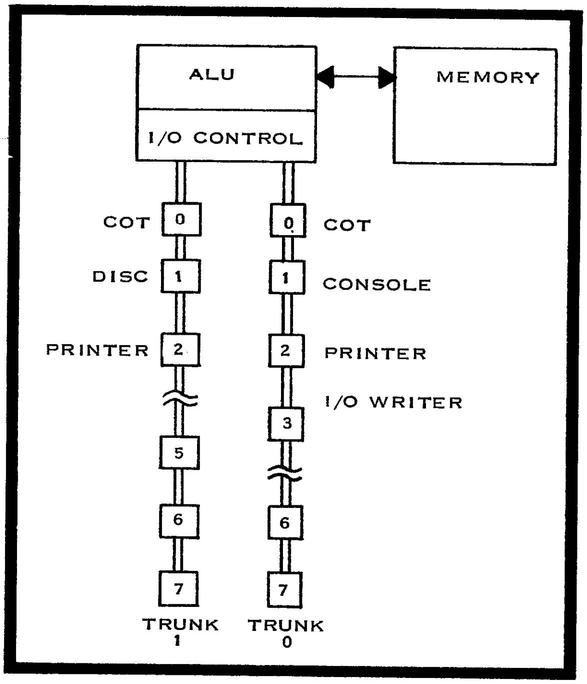
1. Card reader or punched tape reader (COT) at position 0 on Trunk 0 and Trunk 1.

2. Operator's console at position 1 on Trunk 0.

3. Line printer at position 2 on Trunk 0 and Trunk 1.

4. Optional integrated I/O Writer at position 3, Trunk 0.

5. Dual-spindle disc unit at position 1 on Trunk 1.



The COT and printer may be selected on either trunk. Either the integrated card reader or the integrated tape reader, but not both, may be used with the system. The integrated printer and its control logic may be optionally omitted in favor of a free-standing printer, but its position cannot be used for other peripherals, including other printers. When a freestanding printer is used, it must occupy one of the open positions on the trunk. Peripherals that will be run concurrently with the integrated disc unit should be connected through trunk 0 for maximum efficiency (assuming these peripherals do not exceed the 40 kb bandwidth). A second dual-spindle disc unit may be connected at position 1, Trunk 1 and use the same integrated controller as the base unit.

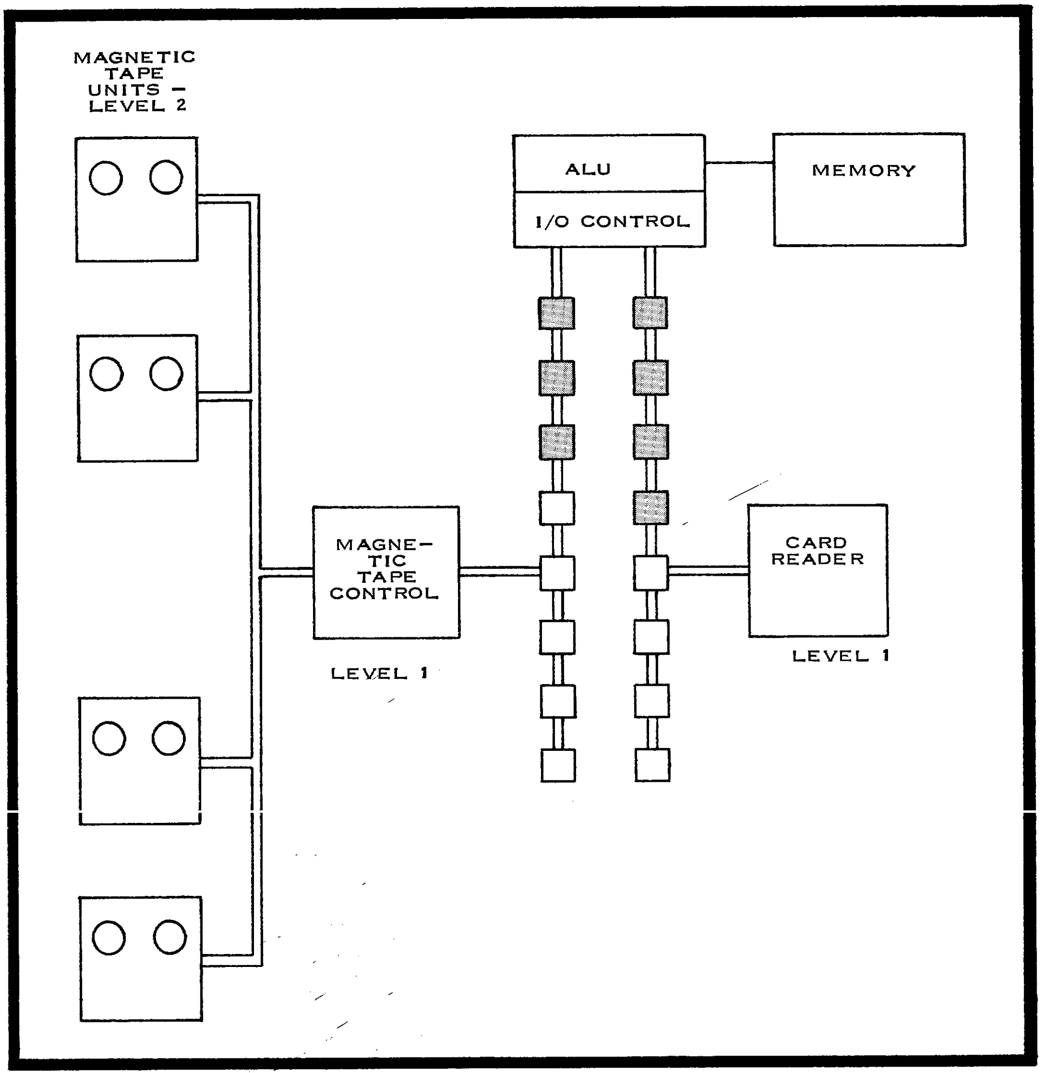
## Freestanding Peripherals

Common trunk or freestanding peripherals may be of two types, depending upon their method of interfacing to the system:

1. Level 1 freestanding peripherals are devices that can interface directly to the common trunk without the interposition of a control unit (controller) or multiplexer. These devices include the controllers and multiplexers themselves and certain low-speed peripherals such as punched tape and card readers.

2. Level 2 freestanding peripherals interface to the I/O trunk through a controller or multiplexer. During I/O operation, the I/O Control communicates with the controller, and the controller supervises the actual peripheral operation by means of a second trunk similar in design to the I/O trunks.

The illustration shows a Level 1 peripheral connected directly to Trunk 0, position 4, and four Level 2 tape handlers connected to Trunk 1, position 4 through a single Level 1 controller:



## FUNCTIONAL OPERATION

I/O Control operation may be divided into three parts: selection, data transfer, and termination.

### Peripheral Selection

An I/O operation is initiated when the ALU selects a peripheral in response to an I/O command. Selection includes choosing the peripheral, initiating the desired function (read, write, print, etc.), and any special instructions required by the specific peripheral. All this information is contained in a variable length field called the Peripheral Address Field (PAF). The address of this field is the effective A operand address after setup of the I/O command.

- Peripheral Address Field (PAF)

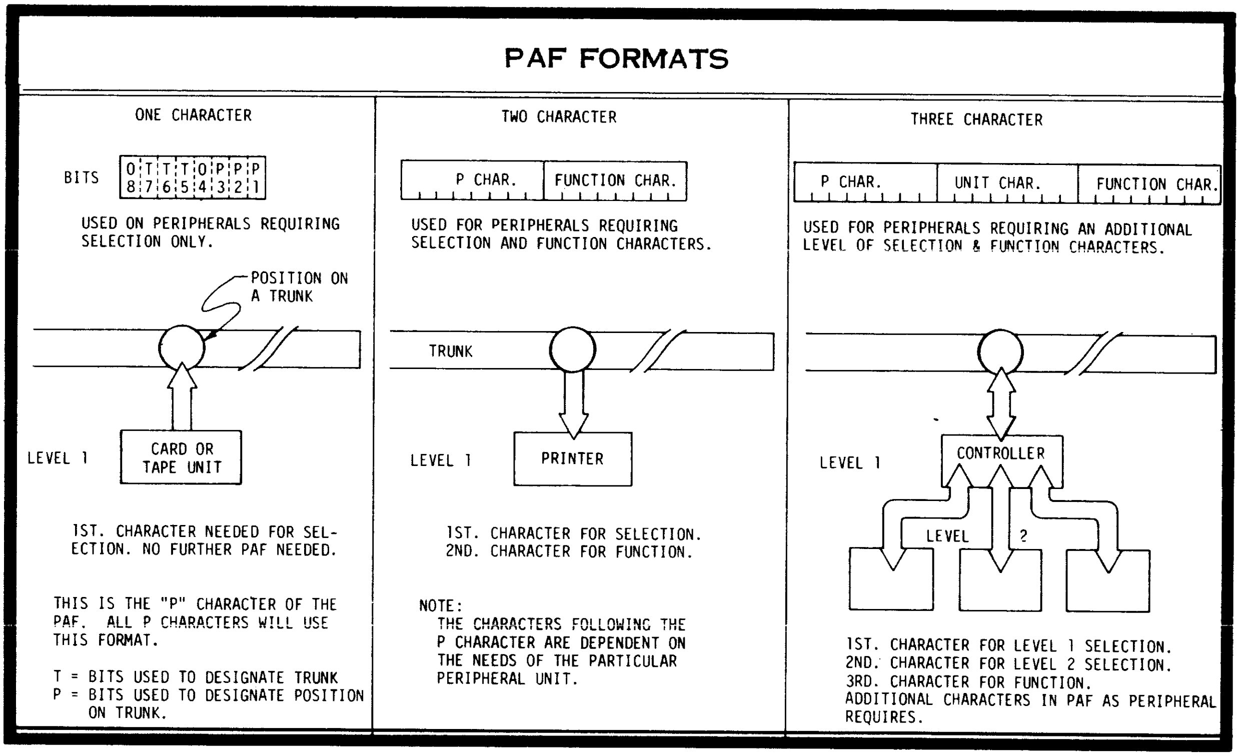
The number of characters in the PAF depends upon the needs of the peripheral to be selected. There is no maximum length to the PAF; the minimum length is one character.

The first character of the PAF is the trunk/position character, referred to as the P character, shown in the following illustration:

| P CHARACTER | | | | | | | |
|---|---|---|---|---|---|---|---|
| b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 |
| 0 | T | T | T | 0 | P | P | P |

TTT = I/O trunk number (0 or 1)
PPP = Position number (0 through 7)

In the NCR Century 100 System, b8 and b4 are always 0. The arrangement of the characters in the PAF following the P character is determined by the functional characteristics of the peripheral. Level 1 freestanding peripherals that perform more than one function but which need no additional addressing require a function code following the P character. Controllers require an additional character in the PAF to select the Level 2 unit connected to the controller.


PAF FORMATS

ONE CHARACTER: USED ON PERIPHERALS REQUIRING SELECTION ONLY.

1ST. CHARACTER NEEDED FOR SELECTION. NO FURTHER PAF NEEDED.

THIS IS THE "P" CHARACTER OF THE PAF. ALL P CHARACTERS WILL USE THIS FORMAT.

T = BITS USED TO DESIGNATE TRUNK
P = BITS USED TO DESIGNATE POSITION ON TRUNK.

TWO CHARACTER: USED FOR PERIPHERALS REQUIRING SELECTION AND FUNCTION CHARACTERS.

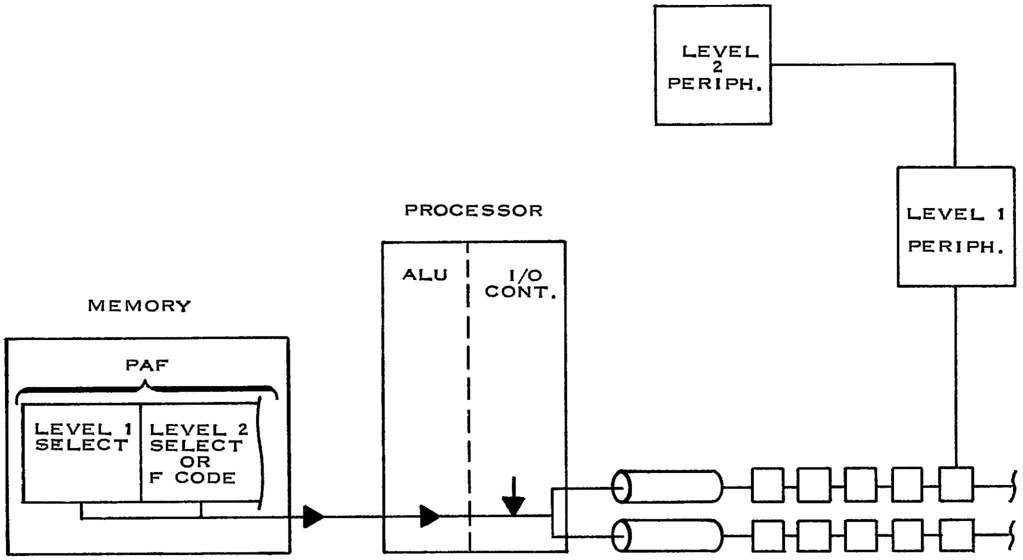
1ST. CHARACTER FOR SELECTION.
2ND. CHARACTER FOR FUNCTION.

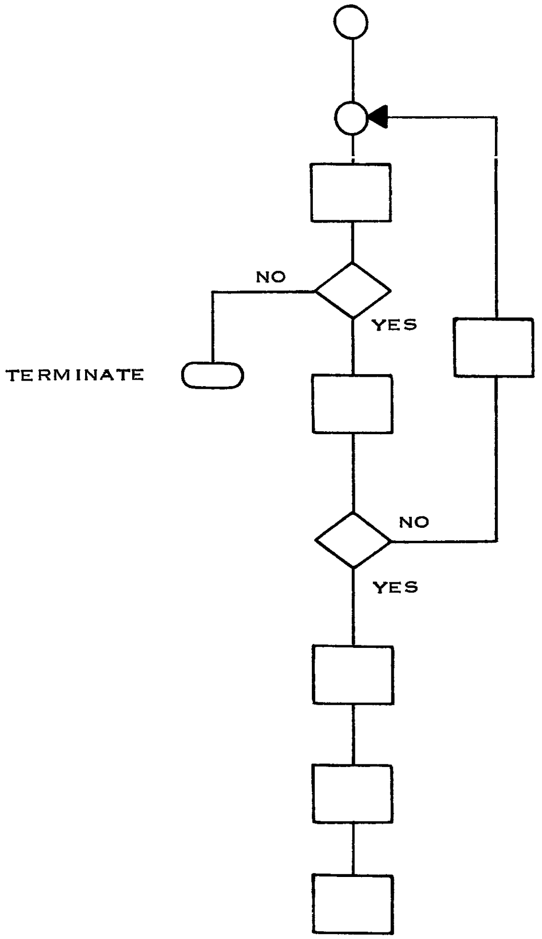
NOTE:
THE CHARACTERS FOLLOWING THE P CHARACTER ARE DEPENDENT ON THE NEEDS OF THE PARTICULAR PERIPHERAL UNIT.

THREE CHARACTER: USED FOR PERIPHERALS REQUIRING AN ADDITIONAL LEVEL OF SELECTION & FUNCTION CHARACTERS.

1ST. CHARACTER FOR LEVEL 1 SELECTION.
2ND. CHARACTER FOR LEVEL 2 SELECTION.
3RD. CHARACTER FOR FUNCTION.
ADDITIONAL CHARACTERS IN PAF AS PERIPHERAL REQUIRES.

# PERIPHERAL SELECTION





I/O COMMAND INITIATED

ACCESS PAF CHARACTER FROM MEMORY

WAS COMMAND RECEIVED BY PERIPHERAL

INCREMENT PAF ADDRESS BY 1

SEND PAF CHARACTER TO PERIPHERAL

IS THIS THE LAST PAF CHARACTER REQUIRED BY THE PERIPHERAL

SET UP ADDRESS FOR S2 CHARACTER STORAGE [EFFECTIVE B OPERAND]

STORE S2 CHARACTER

PROCESSOR TERMINATES I/O COMMAND AND LEAVES I/O CONTROL IN CHARGE OF DATA TRANSFER, PROVIDED SELECTION WAS SUCCESSFUL.

- S2 Status Character

The peripheral responds to attempted selection by transmitting certain control characters to the ALU. These characters, in turn, cause the ALU to generate and store an 8-bit character, called an S2 character, in the address specified by the B operand of the INOUT command. System software examines this S2 character to determine the current state of the peripheral and initiates whatever action is required.

There are four possible S2 bit configurations:

1. Busy (10000000)

   This configuration is stored if a busy indication is detected during selection. The busy indication means that either the I/O trunk is tied up with another peripheral or the selected peripheral itself is busy. If the peripheral or the trunk responds with a Busy status, the INOUT command terminates and software stacks the I/O operation on a list for automatic retry as soon as the trunk is free.

   The common trunk philosophy specifies that random access peripherals such as CRAM or disc can share "seek" time (the time required to locate the track where the information to be stored or read is located). This means that the controller unit for a group of such peripherals can suppress the Busy status and permit initiation of a seek operation even though another peripheral in the group is engaged in a seek, read, or write operation.

2. Standby (10000010)

   This configuration is stored if the peripheral has been put in a Standby condition. A peripheral is placed in Standby whenever the STOP button on its console is pressed. The INOUT command terminates at the end of the current operation on receipt of a Standby code; the software executes a WAIT command, and a console display informs the operator that the unit is in Standby.

3. Inoperative (00000010)

   Two conditions can cause an Inoperative status code to be stored. In one instance, the character is stored because the peripheral is physically inoperative (full stacker, out of media, etc.). An Inoperative status code is also stored if, for any reason, the peripheral does not respond to selection within the allotted P-count time (called a Response Error). In either case, the INOUT command terminates; software executes a WAIT command, and the appropriate console display informs the operator.

4. Command Initiated (01000000)

   This configuration is stored as soon as the selected peripheral has accepted all the PAF characters. When the Command Initiated code is received, the ALU returns to internal processing operations, leaving the I/O Control to supervise data transfer.

## Data Transfer

When the selected peripheral is ready to receive or transmit a character of data, it sends a service request to the processor. The ALU then transfers memory use to the I/O Control at some logical point in processor operation, depending upon the timing sequence of the program. I/O Control uses the available memory cycles to store a data character received from the peripheral or to read a data character out of memory for transfer to the peripheral.

Each service request is accompanied by a response number that is wired into the peripheral's logic circuitry at the time of its installation. The I/O Control uses this response number to calculate the memory address of the control word for the particular peripheral. This control word is an 8-byte field used by the I/O Control to locate data for transfer, to terminate the I/O operation, and to store an S3 or S4 status character. Control words are stored in memory, beginning at location 01024. The format of the control word is shown in the accompanying illustration. The integrated printer on the NCR Century 100 has a special control word of its own that is discussed in the separate publication dealing with this peripheral.

CONTROL WORD FORMAT

| | NA | | | TA | | | |
|---|---|---|---|---|---|---|---|
| S | N3 | N2 | N1 | T2 | T1 | X | X |

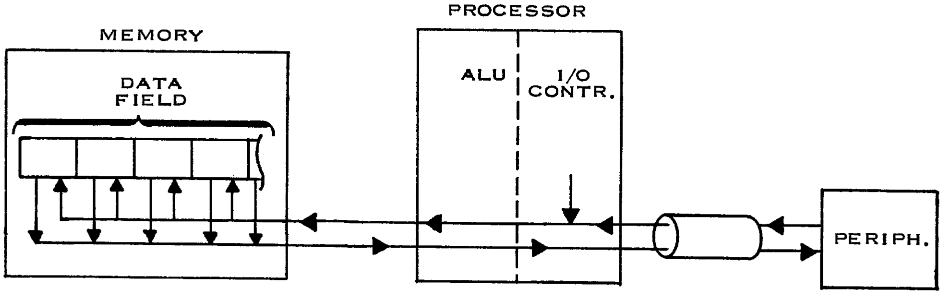
S = S3 or S4 status character (see Termination).

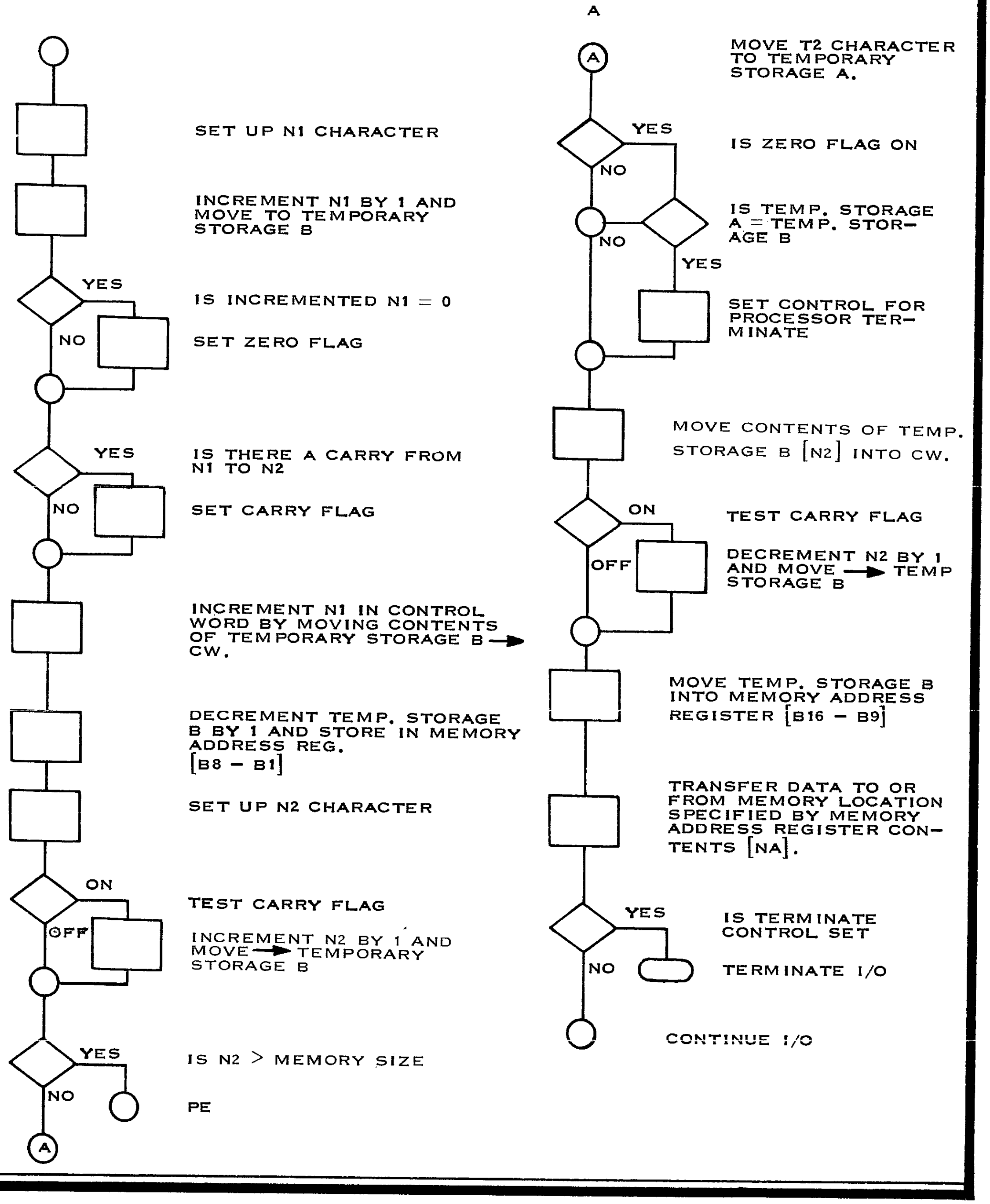
NA = A 3-byte field containing the address of the next character to be accessed by I/O Control for output to the peripheral or the address where the next character received from the peripheral is to be stored. NA is incremented by 1 as each character is transferred.

TA = A 2-byte field containing the terminating address for data readout or storage. Only the four least significant bits of T2 are considered. When these equal the four least-significant bits of N2, and N1 equals 0, the I/O Control knows that the data area specified has been exhausted and sends a processor terminate signal to the peripheral.

XX = A 2-byte field available for software use.

# DATA TRANSFER AND I/O TERMINATION




SET UP N1 CHARACTER
INCREMENT N1 BY 1 AND MOVE TO TEMPORARY STORAGE B
YES
IS INCREMENTED N1 = 0
NO
SET ZERO FLAG
YES
IS THERE A CARRY FROM N1 TO N2
NO
SET CARRY FLAG
INCREMENT N1 IN CONTROL WORD BY MOVING CONTENTS OF TEMPORARY STORAGE B → CW.
DECREMENT TEMP. STORAGE B BY 1 AND STORE IN MEMORY ADDRESS REG. [B8 – B1]
SET UP N2 CHARACTER
ON
TEST CARRY FLAG
OFF
INCREMENT N2 BY 1 AND MOVE → TEMPORARY STORAGE B
YES
IS N2 > MEMORY SIZE
NO
PE
A
A
A
MOVE T2 CHARACTER TO TEMPORARY STORAGE A.
YES
IS ZERO FLAG ON
NO
IS TEMP. STORAGE A = TEMP. STORAGE B
NO
YES
SET CONTROL FOR PROCESSOR TERMINATE
MOVE CONTENTS OF TEMP. STORAGE B [N2] INTO CW.
ON
TEST CARRY FLAG
OFF
DECREMENT N2 BY 1 AND MOVE → TEMP STORAGE B
MOVE TEMP. STORAGE B INTO MEMORY ADDRESS REGISTER [B16 – B9]
TRANSFER DATA TO OR FROM MEMORY LOCATION SPECIFIED BY MEMORY ADDRESS REGISTER CONTENTS [NA].
YES
IS TERMINATE CONTROL SET
NO
TERMINATE I/O
CONTINUE I/O

## Termination

When data transfer is started, the N1 character of NA is incremented by 1 for each byte transferred. When N1 equals 0 (all bits reset by repeated incrementation), the N2 character is incremented by 1 and the four least significant bits (b4-b1) of N2 are compared to the four least significant bits of T2. If they are equal, then the data field has been exhausted. At this point, a processor terminate condition is set up; the last data character is processed and the I/O operation is terminated.

If the bits are not equal, data transfer continues. NA is incremented by 1 for each byte transferred until N1 again equals zero. The N2 character is again incremented and compared to T2. This process continues until N2 and T2 are equal, at which time the last character is processed, the I/O operation is terminated, and the Terminating Indicator is turned on.

- Terminating Indicator

  The peripheral responds to the processor termination signal by requesting service and transmitting an S3 status character along with a terminating status signal. As soon as the ALU transfers supervision to the I/O Control, the I/O Control stores the S3 character in the control word and turns on the Terminating Indicator (TI). In the user state, the processor tests the TI before accessing each command. If the TI is on, the processor enters the BCT flow, turns the S flag ON and stores the address 00264 in the supervisor sequence control register. This means that the next time the ALU enters the command setup phase, the combination of the S flag and the address will send it to the location of a special branch command to a routine for handling I/O termination. During this routine, software examines the S3 character and takes whatever action is required.

If an I/O operation is complete before a processor terminate is received, the peripheral itself requests service and sends a terminating status signal and the appropriate S3 status character. As soon as the I/O Control takes charge from the ALU, it stores the status character in the control word and turns on the TI.

If the I/O Control detects a transmission error during data transfer, it sets the following sequence of operations in motion:

1. The NA is stored in the control word.

2. An error signal is sent to the peripheral or controller.

3. A special S4 status character is stored in the S byte of the control word.

4. The character in error is stored in memory.

The peripheral or controller responds to the error signal by deselecting itself (and any unit it controls, in the case of a controller) when the signal is received.

- S3 Status Character

  The S3 Status Character, transmitted at I/O termination, informs the processor of the results of the I/O operation. Each bit in the character indicates a specific condition by being set to 1.

1. Operation Complete (00XXXXXX)

   This configuration is stored if the I/O operation has been completed; errors and exceptions encountered during the operation are indicated by various combinations of b6 through b1.

2. Segment Complete (11XXXXXX)

   This configuration is sent to the I/O Control when a real-time peripheral controller receives a processor terminate signal before one of the peripherals it controls has finished transmitting data. The processor must then send a read or write function to activate the remote peripheral. If a data character arrives at the controller before the read or write function code is received, a program overload occurs.

3. Error (XX100000)

   This configuration occurs when the selected peripheral detects an error (normally a parity error) during an I/O operation.

   If the error is detected while the peripheral is performing a read operation, it is noted and sent as a bit in the S3 character when terminating status is sent. If the error is detected while writing, a terminating status signal and the proper S3 character are sent to the processor immediately.

4. System Overload (XX010000)

   A system overload is detected by the selected peripheral when the I/O Control does not respond to the unit's request for service within its character time (character time, which is the amount of time required for a peripheral to receive or transmit one byte of data, varies with the individual peripheral unit).

   If a system overload is detected during a write operation, a terminating status signal and the appropriate S3 character are sent to the processor immediately. When a system overload is detected during a read operation, the peripheral notes the condition and sets the proper bit in the S3 character when a terminating status is sent. Data transmission ceases when a system overload is detected.

5. Media (XX001000)

   This configuration is stored whenever the selected peripheral has detected a warning marker, such as a magnetic tape Destination Warning Marker, during a write operation. The warning mark is noted and sent as a bit in b4 of the S3 character with the terminating signal. The I/O control continues data transmission even though the peripheral has detected the marker.

6. Write Lockout or Program Overload (XX000100)

   - Write Lockout -- The configuration is stored when the peripheral attempts to write but is in the Write Lockout state. The elapsed time between S2 and S3 storage in this case may be so slight as to be undetectable by the program.

   - Program Overload -- A Program Overload condition implies that the program was not in a position to accept data from a real-time peripheral when that peripheral transmitted data to the processor; one or more characters are lost. This condition usually occurs after a Segment Complete termination when the program has not had enough time to assign an input area to the next segment.

     This configuration is not stored at the time the condition is detected, but subsequently, when the peripheral is accessed by an I/O command.

7. Inoperative (XX000010)

   The Inoperative configuration is stored when certain malfunctions are detected by the peripheral after it has been activated (out-of-media, torn punch tape, etc.). Data transmission ceases immediately and the terminating status signal is sent along with the status character.

8. Special (XX000001)

   This configuration is stored to indicate any conditions not included above; for example, attempting to write a record on the integrated disc that is longer than a sector length.

9. Transmission Error (10000001)

   This configuration is stored if a transmission parity failure is detected by a peripheral during the I/O operation. When a parity error is detected, the peripheral immediately deselects itself and sends the character.

10. Standby (10000010)

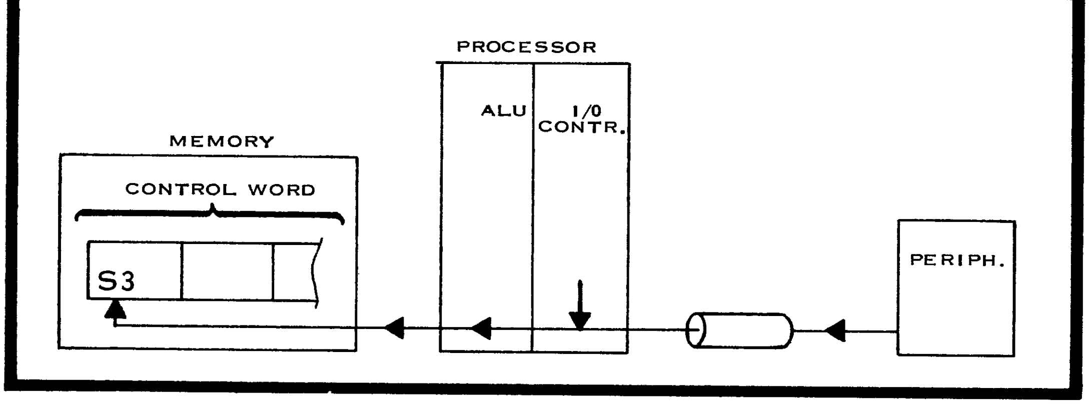
    This configuration is stored if the STOP switch on the selected peripheral has been pressed.

Certain peripherals can have more than one status condition occur during an I/O operation. These conditions are noted by the peripheral's internal logic and are reflected by setting the appropriate S3 character bit before transmission.

- S4 Status Character

  If I/O Control detects a transmission error (parity error), it signals the peripheral to stop sending data and inhibits the transmission of an S3 character. The I/O Control then generates its own status character, called an S4 character, which has the same configuration as the S3 Transmission Error character (10000001), and stores it in the appropriate control word location.

S3 STATUS CHARACTER TRANSMISSION

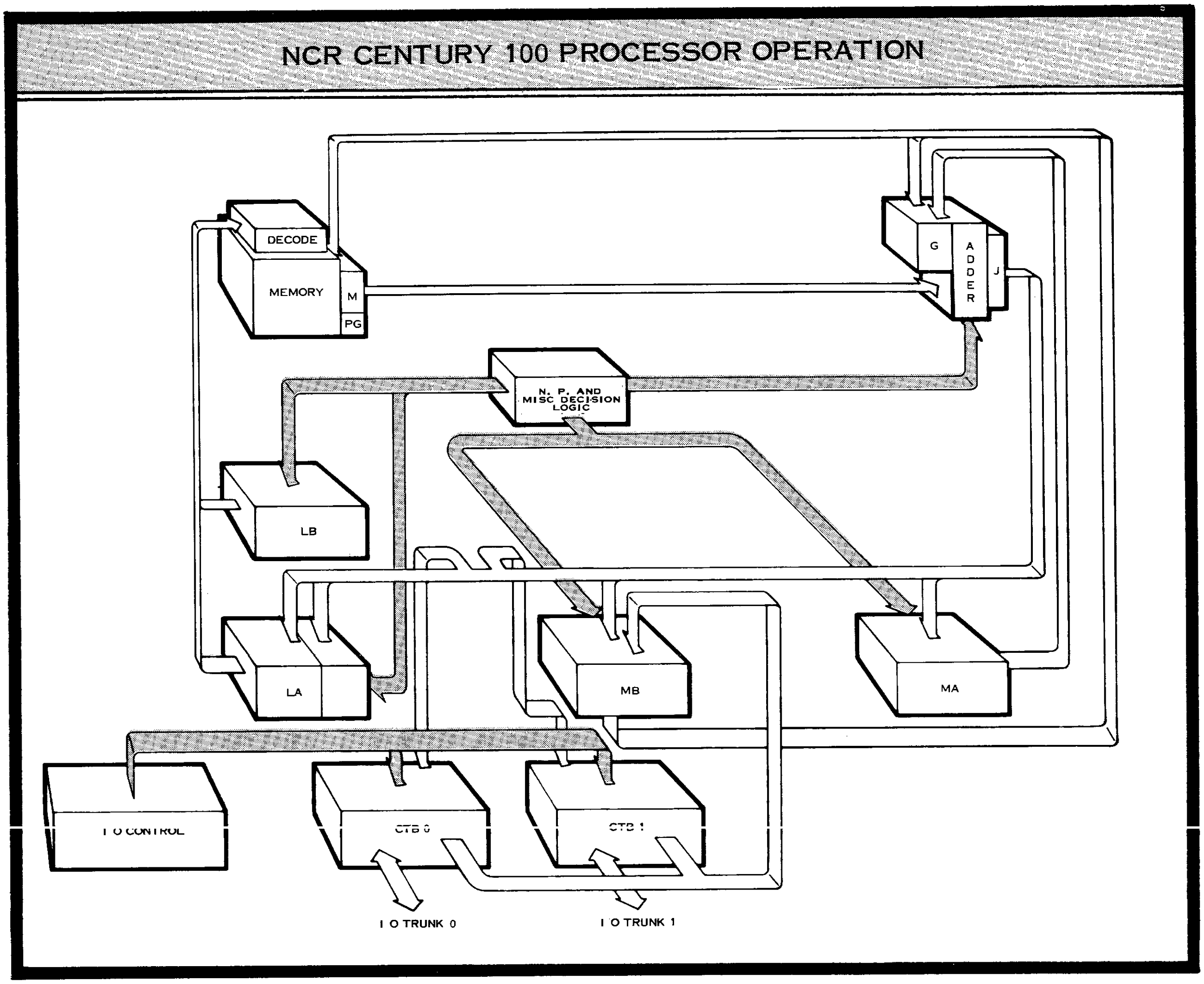


When the S3 or S4 status character is stored, I/O Control sets the I/O Termination Interrupt indicator. Anytime the indicator is ON, the processor traps to an I/O Termination Interrupt routine. In this routine, software examines the S3 or S4 character and takes appropriate action. If the status character indicates the occurrence of any condition requiring operator intervention, system software executes a WAIT command and initiates a console display. The displays, their meaning, and the operator action required in each case are explained in detail in the NCR Century OPERATORS INFORMATION MANUAL.

System Errors

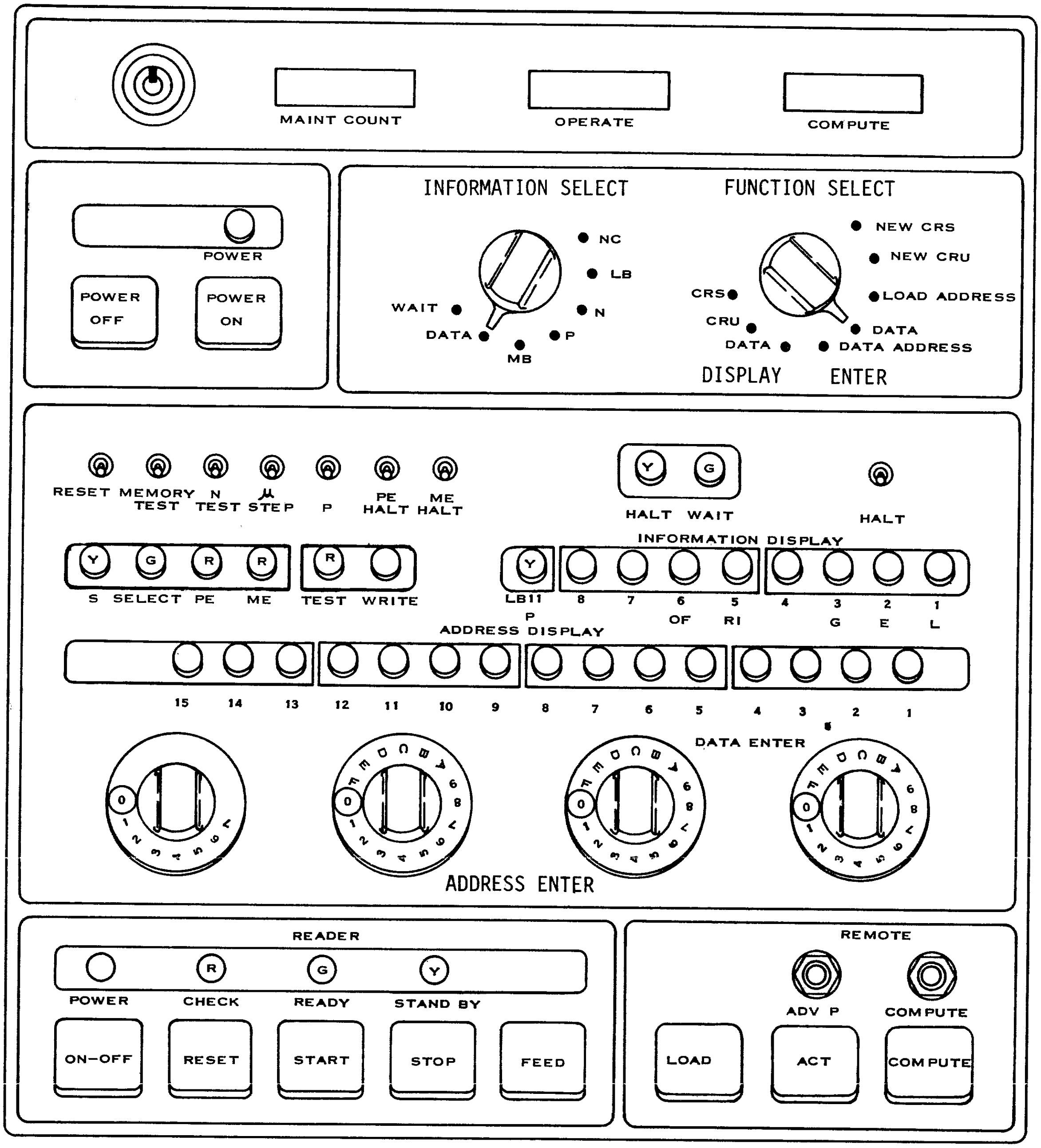
Certain errors or combinations of errors require the operator to reset the system to some starting point in order to clear the condition completely (see the NCR Century OPERATORS INFORMATION MANUAL). The operator presses the RESET switch on the console to send a "clear" signal to all peripherals, which respond by:

1. Deselecting as quickly as possible without waiting to service any existing I/O request or sending any signals (including status) to the processor.
2. Resetting to their initial starting condition (this action does not disturb any data being transmitted to the processor at the moment of reset).



LEGEND

| | |
|---|---|
| M | - MEMORY DATA REGISTER |
| PG | - PARITY GENERATOR |
| G | - ADDER INPUT REGISTER |
| J | - ADDER OUTPUT REGISTER |
| LA, LB | - MEMORY ADDRESS REGISTERS |
| MA, MB | - TEMPORARY DATA STORAGE |
| CTB | - COMMON TRUNK BUFFER |
| ⇨ | CONTROL |



## OPERATOR'S CONSOLE AND GENERAL OPERATING PROCEDURES

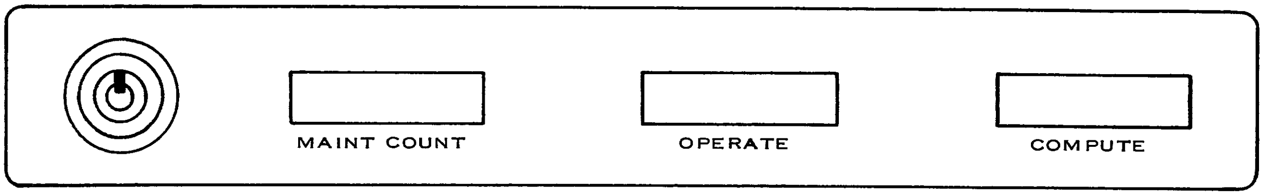
### INTRODUCTION

The operator's console is the control center for system operation. Console switches and indicators operate through position 1 of Trunk 0. The console provides a means of communication between the operator and the computer, permitting him to display the contents of certain registers and memory locations and the condition of various program flags and indicators. The console also serves as a medium through which the operator can alter the contents of certain addresses and respond to system software and user program messages.

The switches, meters, and indicators used by the operator are described in detail in the following sections. Included with the description are the general procedures for console operation. For a more detailed description of system operation, refer to the NCR Century OPERATORS INFORMATION MANUAL.

## TIME METERS AND MAINTENANCE LOCK

The time meters at the top of the console measure the processor's operating, computing, and maintenance times. During maintenance periods, the COMPUTE time meter is turned off by maintenance personnel.



OPERATE

The OPERATE counter registers the total amount of time that the system is operational, whether a program is being run or the system is waiting in a Halt state. On this and the other meters, time is measured in 0.1 hour (6 minute) intervals.

COMPUTE

The COMPUTE counter records the time that the processor is in a run condition; that is, the power is ON, the processor is not in a Halt state, and the maintenance switch is in the operate position. This counter does not record run time when the maintenance switch is in the maintenance position.

MAINT COUNT

The MAINTenance COUNTer records the number of times the maintenance switch is turned to the maintenance position.

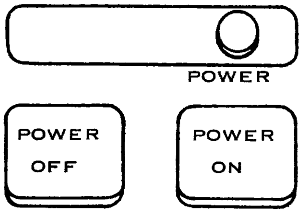
Maintenance Lock

The maintenance lock is a key-operated switch accessible only to NCR service personnel. The switch is used to deactivate the COMPUTE counter and activate certain console switches for maintenance.

## SWITCHES

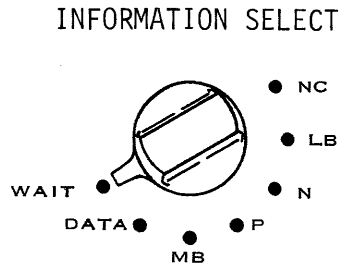
### Power Switches

The POWER ON and POWER OFF switches control 115 volt AC power to the processor only; however, the processor must be ON before any integrated peripherals may be turned on.



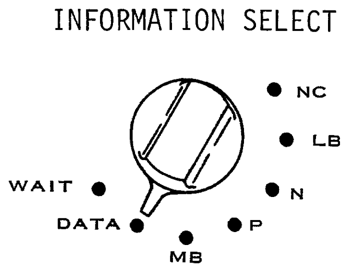
### INFORMATION SELECT Switch

The INFORMATION SELECT switch is a 7-position switch that has two positions available for operator use and the remaining five reserved for maintenance purposes only.



#### ● WAIT Position

The WAIT position, the normal position for the INFORMATION SELECT switch, is used with INFORMATION DISPLAY lights 1-8 to display the wait code of a software or user program message (see the NCR Century OPERATORS INFORMATION MANUAL for a description of wait displays).

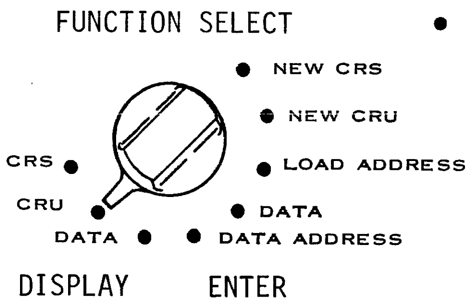


#### ● DATA Position

The DATA position is used to display the remainder of the software or user program message. When the INFORMATION SELECT switch is in this position, two (hex) characters of the message are displayed each time the operator presses the ACT switch. During this operation, the FUNCTION SELECT switch must also be in the DATA (DISPLAY) position.

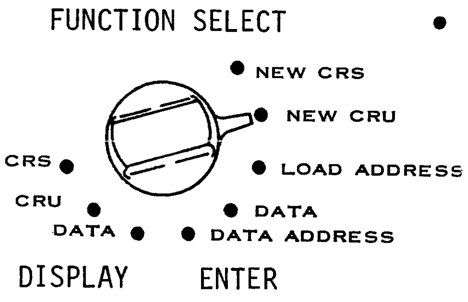
### FUNCTION SELECT Switch

The 8-position FUNCTION SELECT switch is used to read a begin-processing address into one of the two sequence control registers, to store data into memory directly from the console, to display the contents of a given memory location, and to enter the starting address for data input through the integrated card or tape reader. This switch may be used only when the processor is in the Halt or Wait states; it cannot be used while a program is running.



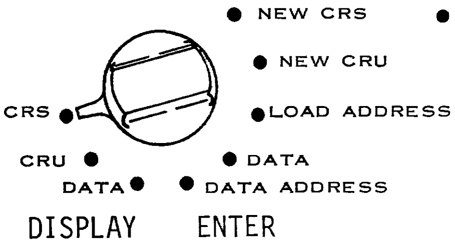
#### ● CRU Position

The operator places the FUNCTION SELECT switch in the CRU position and presses the ACT switch to display the contents of the user sequence control register. This register contains the address of the next command in sequence when the processor is operating in the user state. The contents of the register are displayed in the ADDRESS DISPLAY lights.



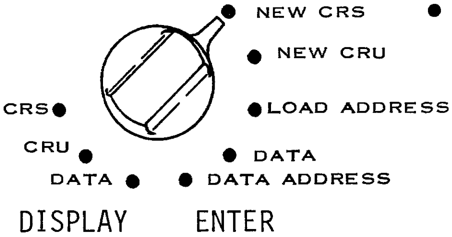
#### ● NEW CRU Position

Placing the FUNCTION SELECT switch in the NEW CRU position permits the operator to enter a new address in the user sequence control register and turn off the S flag by setting the new address on the ADDRESS ENTER switches and pressing the ACT switch.



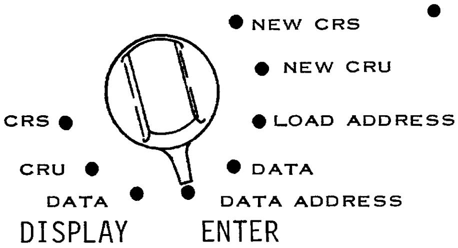
- CRS Position

By setting the FUNCTION SELECT switch to the CRS position and pressing the ACT switch, the operator can display the contents of the supervisor sequence control register on the ADDRESS DISPLAY lights.



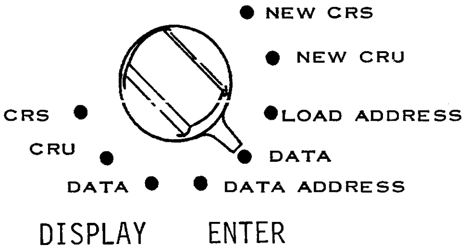
- NEW CRS Position

Setting the FUNCTION SELECT switch to the NEW CRS position and pressing the ACT switch stores an address set up on the ADDRESS ENTER switches in the supervisor sequence control register and turns the S flag ON.



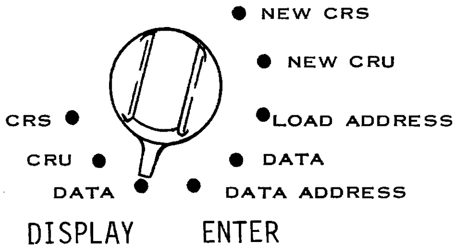
- DATA ADDRESS Position

When the FUNCTION SELECT switch is in the DATA ADDRESS position, the operator can enter an address from which data is to be displayed or into which data is to be stored by setting up the address on the ADDRESS ENTER switches and pressing the ACT switch.



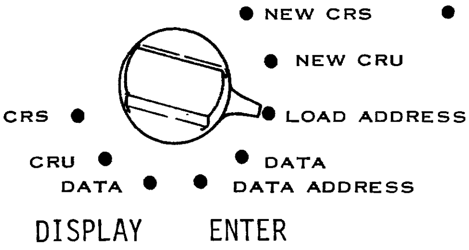
- DATA Position (ENTER Side of Switch)

When the proper address has been entered, as described under DATA ADDRESS, the operator sets the FUNCTION SELECT switch to the DATA (ENTER) position, sets the two DATA ENTER switches to the correct hex representation, and presses the ACT switch to store characters in memory from the console. As each character is input, the storage address is automatically incremented by 1.



- DATA Position (DISPLAY Side of Switch)

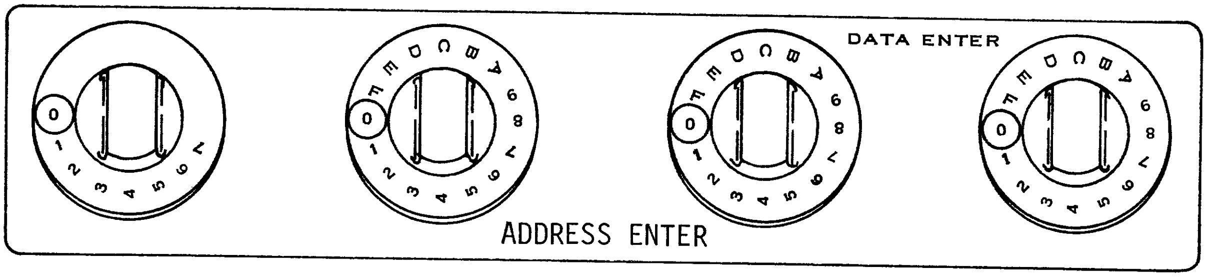
When the proper address has been entered, as described under DATA ADDRESS, the operator sets the FUNCTION SELECT switch to the DATA (DISPLAY) position and the INFORMATION SELECT switch to the DATA position and presses the ACT switch to display the contents of the address in hex notation on the INFORMATION DISPLAY lights. Each time the ACT switch is pressed, the address accessed is incremented by 1.



- LOAD ADDRESS Position

The LOAD ADDRESS Position is used to enter the beginning memory address for data to be input through the integrated COT. With the FUNCTION SELECT switch in this position, the operator sets the ADDRESS ENTER switches to the desired address and presses the ACT switch.

## ADDRESS ENTER Switches



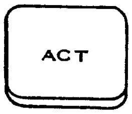
There are four ADDRESS ENTER switches, used in conjunction with the LOAD ADDRESS and DATA ADDRESS positions of the FUNCTION SELECT switch. The three least-significant switches are labelled in hexadecimal from 0 - F; the most-significant switch is labelled 0 - 7, so that the four switches are sufficient to address the entire 32,768 characters of memory in the maximum NCR Century 100 system.

- With the FUNCTION SELECT switch in the DATA ADDRESS position, the four ADDRESS ENTER switches are used to select the memory location from which data is displayed or into which data is stored.
- With the FUNCTION SELECT switch in the LOAD ADDRESS position, the ADDRESS ENTER switches are used to select the beginning address for storage of data read on the COT.

## DATA ENTER Switches

The two least-significant switches in the ADDRESS ENTER set are also labelled DATA ENTER. These switches are used to set up the hexadecimal representation of a character to be stored in memory through the console. The switches are also used to respond to software or user program messages.

## ACT Switch

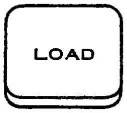


The ACT switch, as explained previously, is used in conjunction with the INFORMATION SELECT and FUNCTION SELECT switches to:

1. Display the contents of a wait message
2. Enter the address of a memory location
3. Display the contents of that location and subsequent locations
4. Display the contents of the CRU or CRS
5. Enter a new address in the CRU or CRS
6. Load an address for storing data from the COT

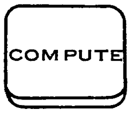
The ACT switch can be used only when the processor is in a Halt state.

### LOAD Switch



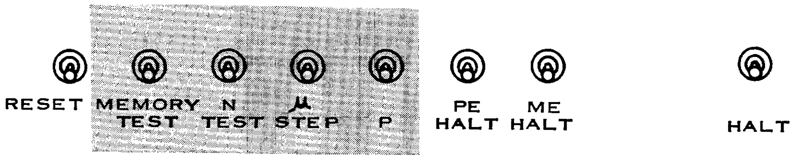
When the beginning address for COT data has been entered, as described under LOAD ADDRESS Position, the operator presses the LOAD switch to start data entry. The LOAD switch can be used only when the processor is in the Halt state (HALT switch ON). When the switch is pressed, the processor enters the supervisor state and the S indicator lights.

### COMPUTE Switch



The COMPUTE Switch is used to initiate program operation at the address stored in the user sequence control register (or supervisor sequence control register if the S indicator is lighted). If the HALT switch is ON, one hardware instruction is executed each time the COMPUTE switch is pressed; if the HALT switch is OFF, the run proceeds without stopping between commands.

### Toggle Switches



There are eight toggle switches on the NCR Century 100 console. Four of these, the PE HALT, the ME HALT, the HALT, and RESET switches, are for the operator's ~~use; the remaining four are interconnected with the maintenance lock and are~~ used by NCR service personnel.

- HALT Switch

  When the HALT switch is set ON (UP position), the processor completes the hardware instruction presently being executed and enters the Halt state. Once the computer has halted in response to the switch setting, the HALT switch must be turned OFF and the COMPUTE switch pressed to return to normal operation.

- RESET Switch

  The RESET switch, which can be activated only when the HALT switch is ON, clears any ME or PE error conditions and deselects all peripherals. This switch is spring-loaded to return to the inactive state as soon as it is released.

- PE HALT Switch

  Setting the PE HALT Switch ON, causes the processor to halt when a PE occurs. Processing may be resumed after turning the PE HALT Switch OFF, but any I/O function that was suspended as a result of the error halt is not completed.

- ME HALT Switch

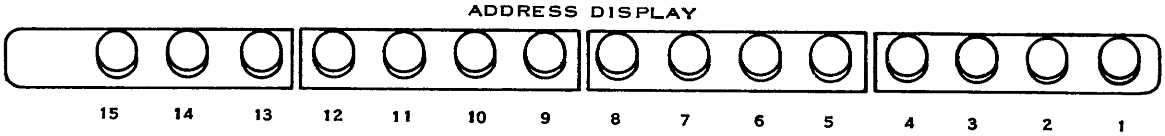
Setting the ME HALT Switch ON, causes the processor to halt when an ME occurs. Processing may be resumed after turning the ME HALT Switch OFF, but any I/O function that was suspended as a result of the error halt is not completed.

NOTE

> The PE HALT and the ME HALT Switches should be used only for debugging programs, since data may be lost if either a PE or an ME occurs during data transmission.

## DISPLAY LIGHTS

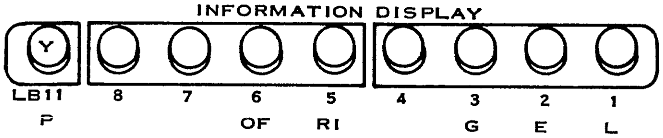
### ADDRESS DISPLAY Lights



ADDRESS DISPLAY lights show the address in the user or supervisor sequence control register when the FUNCTION SELECT switch is set to the CRU or CRS position and the ACT switch is pressed.

These lights also display the address of a memory location into which data is stored through the console or from which data is displayed on the INFORMATION DISPLAY lights.

### INFORMATION DISPLAY Lights



INFORMATION DISPLAY lights are used to display wait messages, the setting of certain flags, or the contents of a memory location addressed through the console.

- Flag Display

Lights 1, 2, 3, 5, and 6 may be used to display the status of the Less Than, Equal To, or Greater Than Flags and the Repeat and Overflow Indicators respectively. To display status, the operator first sets the FUNCTION SELECT switch to the DATA ADDRESS position, sets the ADDRESS ENTER switches to 0004 or to 0014, and presses the ACT switch. He then turns the FUNCTION SELECT switch to the DATA (DISPLAY) position and presses the ACT switch a second time. Each indicator will light if its corresponding flag is ON.

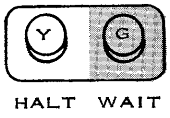
- LB 11/P Display

  The left-most light, labelled LB 11/P, indicates the parity bit setting whenever the contents of a memory location are displayed.

## INDICATORS

There are nine indicators on the NCR Century 100 console, seven of which are used during normal operation, with the remaining two reserved for maintenance purposes.

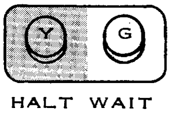
### HALT Indicator



The HALT Indicator lights whenever the processor is in a Halt state. The processor enters a Halt state:

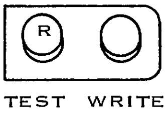
1. When power is first turned on
2. When the hardware detects an unrecoverable error
3. When the operator uses the HALT switch.

### WAIT Indicator



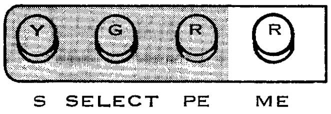
The WAIT indicator lights whenever the processor enters the Wait state to permit the software or user program to display a console message.

### TEST and WRITE Indicators



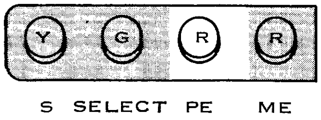
The TEST and WRITE indicators are for maintenance use only.

### ME Indicator



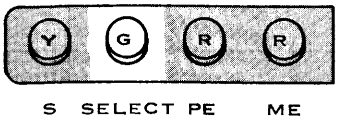
The ME indicator lights whenever a memory error that cannot be corrected by software is detected.

### PE Indicator



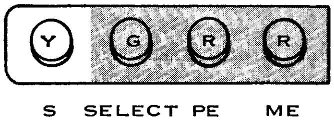
The PE indicator lights whenever a program error that cannot be corrected by software is detected.

### SELECT Indicator



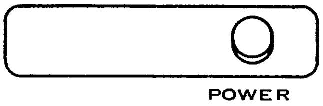
The SELECT indicator lights when the console is selected by the software or user program to display a message for the operator. The operator must respond with an input, using the DATA ENTER switches and the ACT switch, or by pressing the COMPUTE switch, depending upon the message. If COMPUTE is pressed, and data entry is mandatory, the SELECT indicator remains lighted.

### S Indicator



The S indicator lights whenever the processor is in the supervisor state. The processor remains in this state for brief intervals when a software routine that may not be interrupted is running. The S indicator lights when the LOAD switch is pressed or when the ACT switch is pressed while the FUNCTION SELECT switch is in the NEW CRS position. The S indicator is turned off when the ACT switch is pressed while the FUNCTION SELECT switch is in the NEW CRU position.

### POWER Indicator



The POWER Indicator remains lighted while the processor is on.

## GENERAL OPERATING PROCEDURES

### Displaying the Contents of a Memory Location

1. Set the FUNCTION SELECT switch to the DATA ADDRESS position.

2. Set the ADDRESS ENTER switches to the desired address in hexadecimal notation (see page 59).

3. Press the ACT switch.

4. Set the FUNCTION SELECT switch to the DATA (DISPLAY) position.

5. Set the INFORMATION SELECT switch to the DATA position.

6. Press the ACT switch:

   a. The contents of the memory location appear on the INFORMATION DISPLAY lights in binary notation, which may then be converted into hexadecimal notation or into an NCR Century code character.

   b. The address of the location appears in the ADDRESS DISPLAY lights.

7. Press the ACT switch once to display the contents and address of each successive memory location after the beginning address.

8. When the required area has been displayed, reset the INFORMATION SELECT switch to the WAIT position before proceeding.

## Hexadecimal to NCR Century Code Conversion

Any character displayed may be converted directly to NCR Century (ASCII) code by using the following chart:

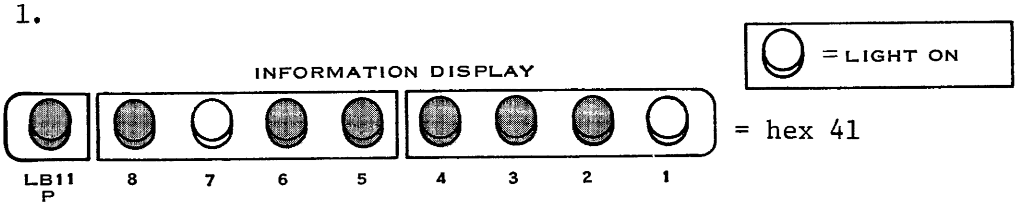
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS | HT | LF | VT | FF | CR | SO | SI |
| 1 | DLE | DC1 | DC2 | DC3 | DC4 | NAK | SYN | ETB | CAN | EM | SUB | ESC | FS | GS | RS | US |
| 2 | SP | ! | " | # | $ | % | & | ´ | ( | ) | * | + | , | − | . | / |
| 3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4 | @ | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 5 | P | Q | R | S | T | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6 | ` | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o |
| 7 | p | q | r | s | t | u | v | w | x | y | z | { | ¦ | } | ~ | DEL |

1. Locate the hex character displayed by the four leftmost INFORMATION DISPLAY lights in the shaded column on the left of the chart.

2. Locate the hex character displayed by the four rightmost INFORMATION DISPLAY lights in the shaded top row of the chart.

3. The equivalent ASCII character is found where the row and column intersect.

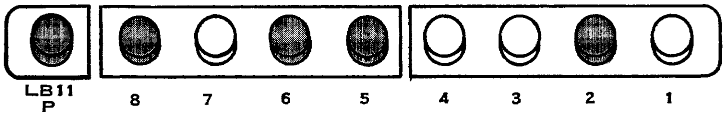
EXAMPLE 1.



= hex 41

4 in the left column and 1 in the top row intersect at the character A in the chart.

EXAMPLE 2.



= hex 4D

4 in the left column and D in the top row intersect at the character M in the chart.

## Entering Data In a Memory Location

1. Set the FUNCTION SELECT switch to the DATA ADDRESS location.

2. Set the ADDRESS ENTER switches to the desired hexadecimal address.

3. Press the ACT switch.

4. Set the FUNCTION SELECT switch to the DATA (ENTER) position.

5. Set the DATA ENTER switches to the hexadecimal representation of the data to be entered.

6. Press the ACT switch.

   a. The address of the memory location appears in the ADDRESS DISPLAY lights.

   b. The data entered is not displayed.

7. Repeat steps 5 and 6 for each subsequent memory location.

## Entering Data From the COT

This procedure is used to enter data, such as the COT boot, before processing begins; it is not used when data is read under software or user program control.

1. Set the HALT switch ON.

2. Set the ADDRESS ENTER switches to the beginning address for data input.

3. Press the ACT switch.

4. Load the media in the COT and place the unit in the Ready state (see Integrated Card or Tape Reader manuals).

5. Press the LOAD switch.

   a. If the card reader is used, one card is read.

   b. If the tape reader is used, the entire length of the tape is read.

   c. The S indicator lights.

## Storing a Starting Address in the CRU or CRS

The procedures for storing a starting address in the CRU and the CRS are the same; only the FUNCTION SELECT switch setting changes. The CRS setting is used to enter the starting address for processing the COT boot after the boot has been read into memory; the CRU setting is used during debug runs and after certain uncorrectable errors.

1. Set the HALT switch ON (UP position).

2. Set the FUNCTION SELECT switch to NEW CRU or NEW CRS.

3. Set the ADDRESS ENTER switches to the hex representation of the desired starting address.

4. Press the ACT switch.

   a. If NEW CRS, the S indicator lights or remains lighted.

   b. If NEW CRU, the S indicator is turned off or remains off.

5. Set the HALT switch OFF.

6. Press COMPUTE to begin processing.

Wait Messages

The procedures outlined here describe the general console operations involved in receiving and responding to wait messages. The many messages generated by system software are beyond the scope of this publication and are explained in detail in the NCR Century OPERATORS INFORMATION MANUAL; user program messages and responses must be provided by the programmer in the system run book.

- Receiving a Wait Message

  When the console WAIT indicator lights:

  1. Make certain that the INFORMATION SELECT switch is in the WAIT position. The first two characters of the message (the wait code) appear automatically in the INFORMATION DISPLAY lights.

  2. Set the INFORMATION SELECT switch to the DATA position.

  3. Set the FUNCTION SELECT switch to the DATA (DISPLAY) position.

  4. Press the ACT switch. The next two characters of the wait message appear in the INFORMATION DISPLAY lights.

  5. Repeat Step 4 until hex characters FF (all lights ON) appear to signal the end of message.

  6. Set the INFORMATION SELECT switch to the WAIT position.

  7. Press COMPUTE to continue processing. If the SELECT indicator lights and the characters FF reappear on the INFORMATION DISPLAY lights, then an operator response is required.

- Responding to a Wait Message Through the Console

  1. Set the FUNCTION SELECT switch to DATA (ENTER).

  2. Set the two DATA ENTER switches to the response indicated in the NCR Century OPERATORS INFORMATION MANUAL or the installation run book.

  3. Press the ACT switch.

  4. Press the COMPUTE switch to terminate the message response and resume processing.

## OPERATE INDICATOR

The console is equipped with an audible signal that calls the operator when the processor enters the wait state or an error halt. The signal does not sound when the processor is operated in the test state. When the signal sounds, it can be stopped by momentarily turning on the HALT switch. The operator can then correct the condition that precipitated the wait or error halt, and can resume processing by pressing the COMPUTE switch.

## OPERATOR MAINTENANCE

There is no operator maintenance required for the processor itself. At the time of installation, however, the user will be provided with a system test routine that checks the functional operation of each unit. Operator maintenance for each integrated peripheral is included in the publication describing that peripheral.

# OPTIONAL I/O WRITER

## INTRODUCTION

An integrated input/output (I/O) writer is available as an optional means of operator-processor communication on the NCR Century 100 System. The I/O Writer permits keyboard entry of data and responses and provides printed output of software and user wait messages. When the writer is included in the system, it operates through an integrated controller located at position 3 on Trunk 0.

## PHYSICAL DESCRIPTION

The I/O writer is installed in the memory cabinet and includes a pin-feed platen on which three-part forms can be typed, an ASCII character keyboard, an audible end-of-line signal, line feed, and a carriage return. The unit has a transfer rate of six characters/second.

## FUNCTIONAL DESCRIPTION

### Modes of Operation

The I/O writer has three functional modes of operation: idle, input, and output.

- Idle Mode

  The idle mode is a neutral state from which the other two modes of operation are entered. The writer is placed in this mode when it is first turned on, when an input or output function terminates, and when a reset function is completed. As long as the unit is in the idle mode, the trunk is free for use by other peripherals.

- Input Mode

  The I/O writer enters the input mode whenever it receives an input permit function code from the processor. While the unit is in this mode, any attempt to access another peripheral on Trunk 0 results in the transmission of a Busy S2 status character. During the input operation, the I/O writer controller receives bits serially from the unit and assembles them in a data register. When a complete byte has been assembled, the controller requests service from the processor to transfer the data character. If additional bits are transmitted by the writer before the character has been transferred, a system overload occurs. If this happens, the program must repeat the selection to recover lost data. Termination of the operation causes an S3 character to be stored; the I/O writer returns to the idle mode as soon as the S3 is received by the processor or if a reset function code is issued at the next selection.

- Output Mode

  When the unit is placed in the output mode by receipt of an output function code, it resets, and the controller requests a data character from the processor. Bits are transferred serially to the data register, assembled into a byte, and transmitted to the writer. As the register empties, the controller requests additional characters, one at a time, until the function terminates and an S3 character is transmitted to the processor. From initiation to termination, the trunk is busy.

Since the controller does not delay transmission following a carriage return character, a programmed wait message must include at least one nonprinting character (such as the line feed control character) between the CR and subsequent characters to ensure proper print positioning.

## Command Operation

- Selection

Input/output operation through the I/O writer requires a two-character PAF to select the trunk and position and transmit a function code

| PERIPHERAL ADDRESS FIELD | |
|---|---|
| 8 7 6 5 4 3 2 1 | 8 7 6 5 4 3 2 1 |
| T T T P P P | F F |

T = Trunk Number (in this case, 000)

P = Position Number (for the I/O writer, a binary value of 3)

F = Function Code. There are three function codes used with the I/O writer. Only the two least-significant bits of the code are used; the remaining six bits are ignored.

**FUNCTION CODES**

| CODE | NAME | FUNCTION |
|---|---|---|
| XXXXXX00 | RESET | TURNS THE INPUT PERMIT FLAG OFF AND PUTS THE INTEGRATED I/O WRITER CONTROL INTO THE IDLE MODE. THE PROCESSOR IS NOT NOTIFIED UNTIL THE FLAG IS TURNED OFF AND THE COMMAND-INITIATED S2 STATUS CHARACTER IS STORED. NO S3 STATUS CHARACTER IS SENT TO THE PROCESSOR. |
| XXXXXX01 | INPUT PERMIT | TURNS THE INPUT PERMIT FLAG ON AND PUTS THE INTEGRATED I/O WRITER CONTROL INTO THE INPUT MODE. THE PROCESSOR IS NOT NOTIFIED UNTIL THE FLAG IS TURNED ON AND THE S2 STATUS CHARACTER IS STORED. NO S3 STATUS CHARACTER IS SENT AT THIS TIME. |
| XXXXXX10 | OUTPUT | PLACES THE INTEGRATED I/O WRITER CONTROL INTO THE OUTPUT MODE. THE PROCESSOR IS NOT NOTIFIED UNTIL THE INTEGRATED CONTROL HAS ENTERED THE OUTPUT MODE AND THE COMMAND-INITIATED S2 STATUS CHARACTER IS STORED. |

- Status Character Transmission

The I/O writer operates within the general frame of status character transmissions described in the I/O CONTROL section of this publication, with three significant exceptions.

During the selection process, the integrated controller transmits an S2 status character to indicate the result of the attempted selection. There are only two S2 characters associated with the I/O writer: command initiated, and inoperative. The I/O writer never stores the S2 busy; if it is busy when the processor attempts to select it, it stores the S2 inoperative instead.

During the I/O operation, just prior to completion, the integrated controller stores an S3 character to indicate the results of the operation. The controller can generate any of the four S3 characters listed in the table below; it never stores the S3 transmission error character, however, since it transmits no parity bit; when it encounters an error condition, it stores the S3 inoperative.

Parity bits for I/O writer characters are generated in the processor's I/O control. This means that the I/O control does not detect transmission errors (parity errors) in characters generated by the I/O writer and does not store the S4 transmission error character. During output operations, the I/O control does detect parity errors, but since such a parity error would be a memory error (ME), it would cause an ME halt.

### INPUT/OUTPUT WRITER STATUS CHARACTERS

| STATUS | CHARACTER | CONDITION | INDICATES |
|---|---|---|---|
| S2 | | | CONDITION DURING SELECTION PROCESS |
| | 00000010 | INOPERATIVE | I/O WRITER CONTROL IS PERFORMING A FUNCTION WHEN A SELECTION ATTEMPT IS MADE; AN EXCEPTION TO THIS IS THAT THE RESET MODE IS ACCEPTED. |
| | 01000000 | COMMAND INITIATED | ALL OPERATING CONDITIONS SATISFIED, PAF ACCEPTED. |
| S3 | | | CONDITION OCCURRING AFTER SELECTION BUT BEFORE TERMINATION |
| | 00000010 | INOPERATIVE | I/O WRITER BECOMES INOPERATIVE AFTER SELECTION BUT BEFORE NORMAL TERMINATION. |
| | 00010000 | SYSTEM OVERLOAD | INTEGRATED I/O WRITER CONTROL DID NOT RESPOND TO THE UNIT'S REQUEST FOR SERVICE BEFORE THE ARRIVAL OF THE NEXT CHARACTER AT THE REGISTER. |
| | 11000000 | SEGMENT COMPLETE | THE NA CHARACTER OF THE CW IS EQUAL TO THE TA CHARACTER OF THE CW ON INPUT OR OUTPUT AND LAST CHARACTER ON INPUT WAS NOT END-OF-MESSAGE. |
| | 00000000 | OPERATION COMPLETE | AN END-OF-MESSAGE [EOM] WAS RECEIVED ON INPUT AND NO ERRORS OR EXCEPTION CONDITIONS OCCURRED. |

- I/O Termination

Normal termination occurs when a processor terminate signal is received or an end-of-message character (EOM) is detected. At processor termination, a segment complete S3 character is transmitted; the operation complete S3 is transmitted for EOM terminations.

When the processor encounters an error condition related to the I/O writer, it signals the I/O writer of the situation by means of the latent error lines in the trunk. When the I/O writer receives this latent error signal, it turns its own input permit flag OFF and enters the idle mode; no S3 character is transmitted. When the controller itself detects an error, it turns its input permit flag OFF, transmits the inoperative S3 character, and then enters the idle mode.

Detection of an EOM character terminates I/O operation during input only, after the controller has transmitted the EOM character to the processor. During an output operation, the EOM is output as a normal character received from the processor.

## DATA FORMAT

The I/O writer's character set is identical to the NCR Century internal character set. Bits 6 and 7 are both set to 0 or 1 in control characters to indicate their control function to the processor.

# NCR CENTURY 100 SPECIFICATIONS

## PHYSICAL SPECIFICATIONS

| CHARACTERISTIC | SPECIFICATION | | | |
|---|---|---|---|---|
| | INTEGRATED SYSTEM | PROCESSOR BAY | POWER SUPPLY | MEMORY |
| AC INPUT | SINGLE PHASE<br>4—WIRE<br>4 AWG<br>120/240 V<br>60 HZ | | | |
| KVA | 8.7 | | | |
| BTU/H | 23, 000 | | | |
| CURRENT BY LEG | 1 = 40.0<br>2 = 40.0 | | | |
| TOTAL CURRENT AT NOMINAL VOLTAGE | TURN—ON = 80<br>AVER. OP. = 61<br>MAX. OP. = 70 | | | |
| CIRCUIT BREAKER | 2—POLE, 70 A | | 2—POLE, 70 A | |
| DIMENSIONS | | | | |
| HEIGHT | 48 INCHES | 48 INCHES | 42 INCHES | 42 INCHES |
| WIDTH | 169 INCHES* | 41 INCHES | 41 INCHES | 34 INCHES |
| DEPTH | 27 INCHES | 27 INCHES | 27 INCHES | 27 INCHES |
| WEIGHT | 3400 POUNDS* | 550 POUNDS | 700 POUNDS | 390 POUNDS |
| ACCESS REQUIREMENTS | | | | |
| FRONT | 36 INCHES | 36 INCHES | 36 INCHES | 36 INCHES |
| REAR | 36 INCHES | 36 INCHES | 36 INCHES | 36 INCHES |
| RH SIDE | 36 INCHES | —— | —— | —— |
| LH SIDE | 36 INCHES | —— | —— | —— |
| CONVENIENCE OUTLETS | 15 A | —— | 15 A | —— |
| AIR FLOW | TOP TO BOTTOM | TOP TO BOTTOM | TOP TO BOTTOM | TOP TO BOTTOM |
| LOGIC CABLE NUMBER | 315—05104—XX | | | |

* DOES NOT INCLUDE 655—102 ADD—ON DISC.

The required environmental conditions for the NCR Century 100 Processor are as shown below.

| OPERATING LIMITS | |
|---|---|
| Temperature | 68° to 78°F Dry Bulb |
| Humidity | 40% to 60% Relative |
| Altitude | 7000 feet maximum |

PRODUCT INFORMATION--NCR CENTURY
PROCESSORS

NCR REFERENCE MANUAL

An Educational Publication

number: 2.1
page: 1 of 47
date: Dec. 69

ST-9402-12
BINDER NO. 0141

# NCR CENTURY 100 HARDWARE COMMANDS AND COMMAND TIMING

## INTRODUCTION

### SCOPE AND PURPOSE OF THE PUBLICATION

This publication contains a functional description of the 19 hardware commands used by the NCR Century 100 System. It is intended as a supplement to the NCR CENTURY 100 PROCESSOR manual, which is prerequisite reading, and is not a substitute for the NEAT/3 manual. All command descriptions assume that:

1. A compiled program resides in memory in a form recognizable to the processor hardware.
2. The command setup phase, described in detail in the NCR CENTURY 100 PROCESSOR manual, has been completed.

Commands are described in terms of preserved values, both at the end of the setup phase and the end of the execution phase.

### COMMAND TIMING

Total command time in machine cycles is calculated as the sum of the setup time (including any indexing) and command execution time. One memory cycle is approximately 800 nanoseconds.

$$M = S + E$$

Where:

$M$ = number of cycles required for setup and execution.
$S$ = number of cycles required for command setup.
$E$ = number of cycles required for command execution.

The quantity S is calculated from the equation

$$S = 10 + 8R_A + 1_A R_A + K(10 + 8R_B + 1_B R_B)$$

Where:

$R_A$ = 1 if an index register is designated for the A operand
= 0 if no index register is designated for the A operand

$K$ = 1 for the two-address mode of the command
= 0 for the one-address mode of the command

$R_B$ = 1 if an index register is designated for the B operand
= 0 if no index register is designated for the B operand

$1_A$ = 1 if incremental indexing is specified for the A operand;
otherwise, $1_A$ = 0

$1_B$ = 1 if incremental indexing is specified for the B operand;
otherwise, $1_B$ = 0

## ADDRESS CONVENTION

The address of the leftmost character of each command must be 0 modulo 4 (i.e., the address must be evenly divisible by 4). Attempting to execute a command that violates this specification results in a PE interruption (see NCR CENTURY 100 PROCESSOR manual).

HARDWARE COMMANDS

| COMMAND | CODE | PAGE |
|---|---|---|
| PACK | 76 | 4 |
| UNPACK | 77 | 10 |
| ADD BINARY | 96 | 15 |
| SUBTRACT BINARY | 97 | 19 |
| ADD UNSIGNED | 98 | 23 |
| SUBTRACT UNSIGNED | 99 | 29 |
| MOVE A RIGHT TO LEFT | 100 | 35 |
| COMPARE BINARY | 101 | 38 |
| REPEAT | 102 | 42 |
| WAIT | 103 | 43 |
| BRANCH OVERFLOW | 104 | 44 |
| BRANCH LESS | 105 | 44 |
| BRANCH EQUAL | 106 | 44 |
| BRANCH LESS OR EQUAL | 107 | 44 |
| BRANCH GREATER | 108 | 44 |
| BRANCH LESS OR GREATER | 109 | 44 |
| BRANCH GREATER OR EQUAL | 110 | 44 |
| BRANCH UNCONDITIONALLY | 111 | 44 |
| INOUT | 112 | 45 |

## SYMBOLS AND NOTATIONS USED IN COMMAND DESCRIPTIONS

C = the decimal equivalent of the binary value of the seven low-order bits of the command code (Q) character. The first representation in parentheses, following the decimal value, is the hexadecimal equivalent of the two-address code; the second representation is the hexadecimal representation of the one-address form of the code.

A = the effective A operand address after command setup and indexing.

B = the effective B operand address after command setup and indexing.

T = the effective T (length) character after command setup.

()= the contents of, after command setup and before command execution:

(B) = the contents of the address designated by the B operand after command setup.

$\underline{X}$ = the value of, after command execution:

$\underline{T}$ = the value of the T character after command execution.

$\underline{B}$ = the value of the B operand after command execution.

$(\underline{B})$= the value of the contents of the address designated by the B operand after command execution.

NOTE

$\underline{T}$ = T, invariably.

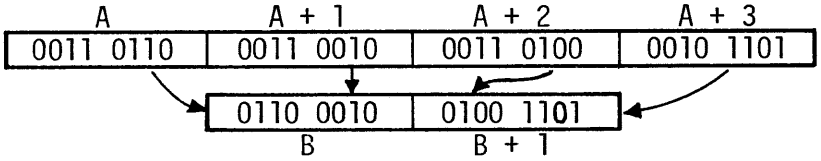
$\underline{B}$ is specified for each command.

Byte representations have been arbitrarily divided into two 4-bit segments as an aid to the reader; no such division exists in memory.

## C = 76(4C)(CC)

The contents of the field specified by the effective A address are packed into the field specified by the effective B address. That is, the least significant four bits (b4-bl) of each character in a pair of A characters are combined to form a single 8-bit character which is stored in one B character.

Packing is performed sequentially, from left to right, starting at the low-order A and B addresses. The four least-significant bits of the left character of the A pair replace the four most significant bits of the B character; the four least significant bits of the right character of the A pair replace the four least significant bits of the B character:



Packing should be performed only on those characters shown in the shaded area on the following chart:

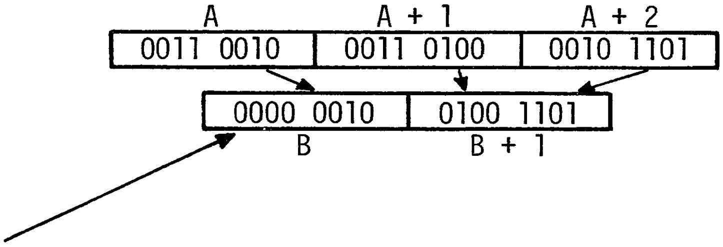
**NCR CENTURY CODE CHART**

| $B_4-B_1$ → <br> $B_8-B_5$ ↓ | | 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 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0000 | 0 | NUL | SOH | STX | ETX | EOT | ENQ | ACK | BEL | BS | HT | 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 | [ | \ | ] | Λ | _ |
| 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 |

These are the only characters that can be restored to their original values by the UNPACK command, as explained on page 10. If any other characters are packed, they lose their original values.

The T portion of the command specifies the number of characters to be packed (length of the A field) and may be any value from 0 through 255, with 0 considered to be 256. The number of B characters affected is T/2 if T is even, and (T + 1)/2 if T is odd. Where T is odd, the leftmost character of the first A pair is treated as the odd character to be right-justified in the left character of the B field; that is, b4-b1 of the A field character are moved to b4-b1 of the B field character, with b8-b5 of the B field character set to zeros.

T odd:



Zeros stored
in most significant
4 bits of leftmost
B character.

When the command is completed, the field that was designated by the effective A operand retains its initial contents; the field that was designated by the effective B operand contains the packed data from the A field.

If the A and B fields overlap, the final results may be different from the results of an operation in which the fields do not overlap. In this case, the initial contents of the A field are altered.

B = B + T/2 if T is even and not equal to zero.*

B = B + (T + 1)/2 if T is odd.

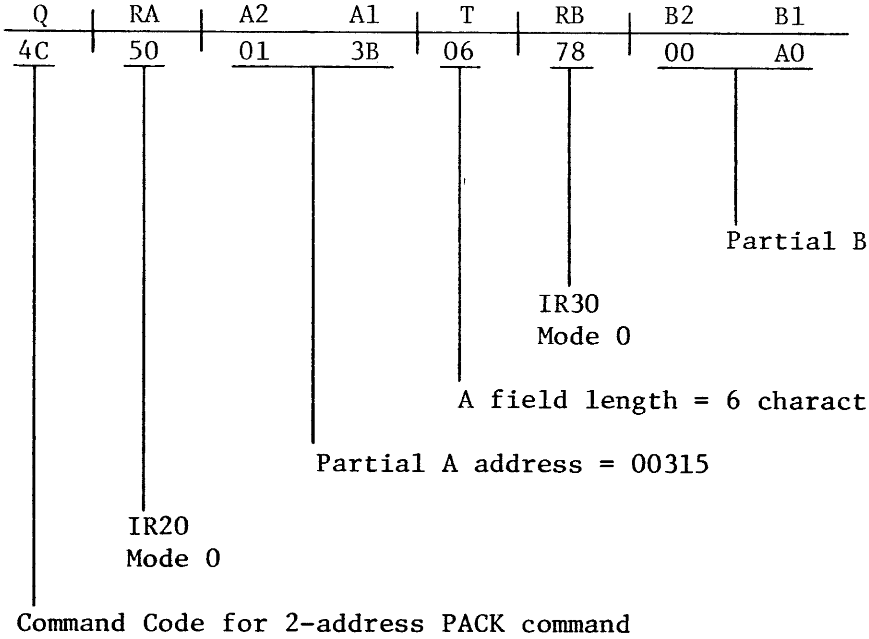
B = T + 128 if T = 0

* That is, the B operand address following command execution is equal to the B operand address following command setup plus one-half the length. This notation will be used throughout the publication.

Command Execution Time

E = (19/2)T + 9, if T is even.
E = (19/2)(T + 1) +9, if T is odd.

Example 1.


Q
RA
A2
A1
T
RB
B2
B1
4C
50
01
3B
06
78
00
A0
Partial B address = 00160
IR30
Mode 0
A field length = 6 characters (bytes)
Partial A address = 00315
IR20
Mode 0
Command Code for 2-address PACK command

(All commands are represented in hexadecimal notation. Refer to NCR CENTURY 100 PROCESSOR manual.)

Assume the index registers contain the following:

(IR20) = 07DO (02000)
(IR30) = OBB8 (03000)

After command setup:

Effective A address = 07DO + 013B = 090B (02315)
Effective B address = OBB8 + OOAO = OC58 (03160)

Before command execution, the A field contains the following information (the contents of the B field are not significant):

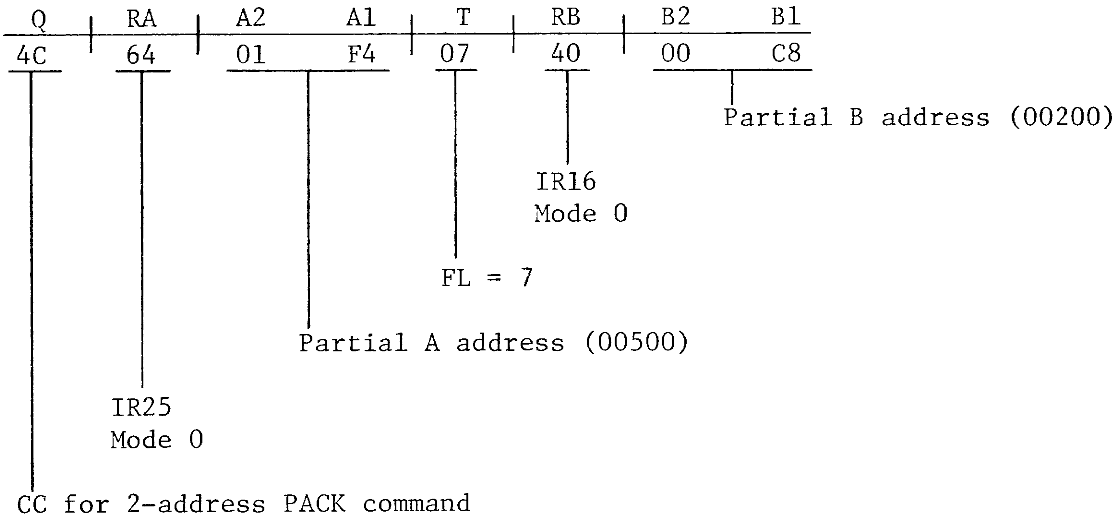
| A | 090B | 090C | 090D | 090E | 090F | 0910 |
|---|---|---|---|---|---|---|
| (A) | 0011 0000 | 0011 0011 | 0011 0010 | 0011 0100 | 0011 1001 | 0010 1101 |

Following command execution, the contents of the A field are unchanged, the contents of the B field are:

| B | 0C58 | 0C59 | 0C5A |
|---|---|---|---|
| $(\underline{B})$ | 0000 0011 | 0010 0100 | 1001 1101 |

$\underline{B}$ = OC58 + 03 = OC5B(03163)

Example 2. (T is odd)


Q
RA
A2
A1
T
RB
B2
B1
4C
64
01
F4
07
40
00
C8
Partial B address (00200)
IR16
Mode 0
FL = 7
Partial A address (00500)
IR25
Mode 0
CC for 2-address PACK command

Assume the index registers contain the following:

(IR16) = 0640 (01600)
(IR25) = 09C4 (02500)

After command setup:

Effective A address = 09C4 + 01F4 = 0BB8 (03000)
Effective B address = 0640 + 00C8 = 0708 (01800)

Before command execution:

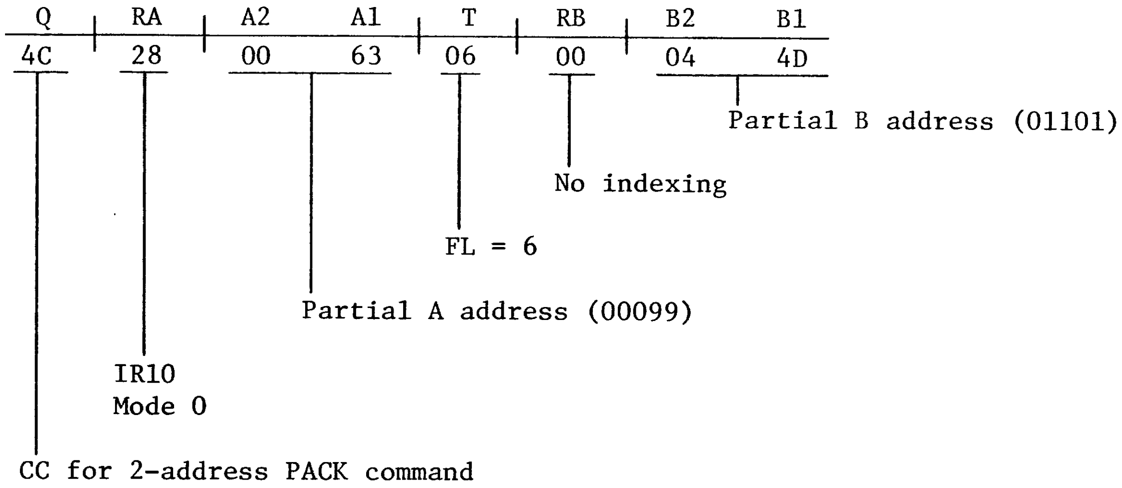
| A | 0BB8 | 0BB9 | 0BBA | 0BBB | 0BBC | 0BBD | 0BBE |
|---|---|---|---|---|---|---|---|
| (A) | 0010 1010 | 0010 0010 | 0011 0010 | 0011 0011 | 0011 0110 | 0011 0101 | 0010 1011 |

After command execution:

| B | 0708 | 0709 | 070A | 070B |
|---|---|---|---|---|
| ($\underline{B}$) | 0000 1010 | 0010 0010 | 0011 0110 | 0101 1011 |

$\underline{B}$ = 0708 + 04 = 070C (01804)

Example 3. (Fields overlap)



| Q | RA | A2 | A1 | T | RB | B2 | B1 |
|---|---|---|---|---|---|---|---|
| 4C | 28 | 00 | 63 | 06 | 00 | 04 | 4D |

Partial B address (01101)

No indexing

FL = 6

Partial A address (00099)

IR10
Mode 0

CC for 2-address PACK command

Assume IR10 contains 03E8 (01000)

After command setup:

Effective A address = 03E8 + 0063 = 044B (01099)
Effective B address = 044D (01101) since no indexing is required.

Before command execution:

This portion of the A field overlaps the B field.

| A | 044B | 044C | 044D | 044E | 044F | 0450 |
|---|---|---|---|---|---|---|
| (A) | 0011 0001 | 0011 0111 | 0011 1001 | 0011 0111 | 0011 0110 | 0011 0101 |

Because the fields overlap, they are altered during command execution as follows:

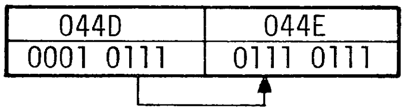
1. b4 - b1 of 044B replace b8 - b5 of 044D:

| 044B | 044D |
|---|---|
| 0011 0001 | 0001 1001 |

2. b4 - b1 of 044C replace b4 - b1 of 044D:

| 044C | 044D |
|---|---|
| 0011 0111 | 0001 0111 |

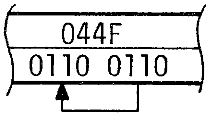
3. b4 - b1 of 044D, which were altered in Step 1, replace b8 - b5 of 044E:



| 044D | 044E |
|---|---|
| 0001 0111 | 0111 0111 |

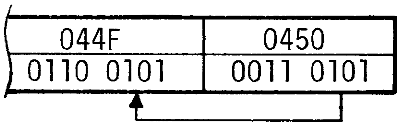
4. b4 - b1 of 044E remain unchanged, since, in effect, they replace themselves.

5. b4 - b1 of 044F replace b8 - b5 of 044F:



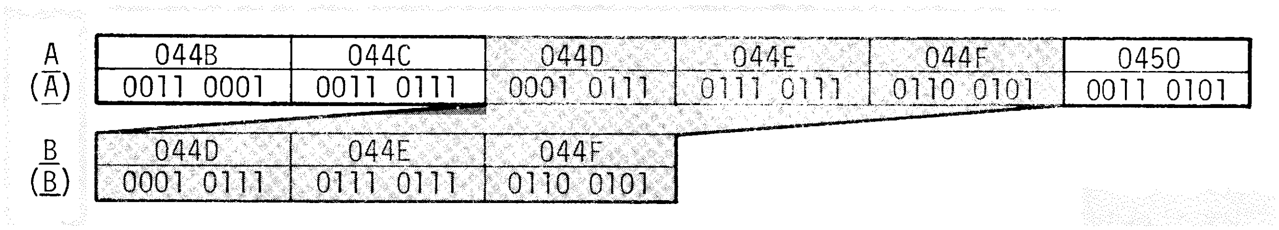
| 044F |
|---|
| 0110 0110 |

6. b4 - b1 of 0450 replace b4 - b1 of 044F:



| 044F | 0450 |
|---|---|
| 0110 0101 | 0011 0101 |

7. Following command execution:



| A | 044B | 044C | 044D | 044E | 044F | 0450 |
|---|---|---|---|---|---|---|
| (A) | 0011 0001 | 0011 0111 | 0001 0111 | 0111 0111 | 0110 0101 | 0011 0101 |
| B | 044D | 044E | 044F | | | |
| (B) | 0001 0111 | 0111 0111 | 0110 0101 | | | |

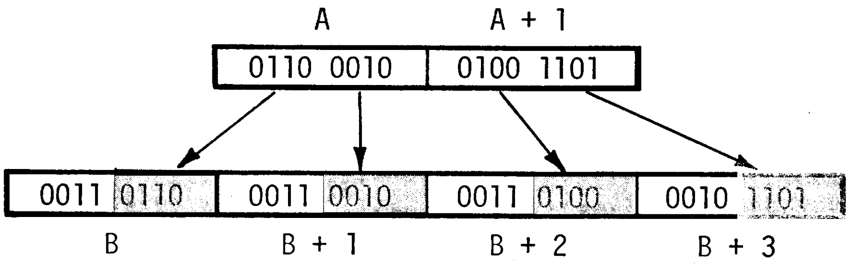
B = 044D + 03 = 0450 (01104).

## C = 77(4D)(CD)

The contents of the field specified by the A operand are unpacked into the field specified by the B operand. That is, each 8-bit A character is divided into two 4-bit digits. Appropriate zone bits are appended to each 4-bit character forming two NCR Century characters. The two characters are then stored in a pair of adjacent locations in the B field.

Unpacking is performed sequentially, from left to right, starting at the low-order A and B addresses. Bits b8 - b5 of each A character replace b4 - b1 of the left B character in each pair, and the appended zone bits are stored in b8 - b5 of the B character. Bits b4 - b1 of each A character replace b4 - b1 of the right B character in each pair, the zone bits again occupying b8 - b5.

If the binary value of the 4-bit A character is nine or less, the zone bits 0011 are appended; if the binary value is greater than nine, the zone bits 0010 are appended:



The T portion of the command specifies the number or characters to be unpacked (length of the field designated by the effective A operand). T may range in value from 0 through 255, with 0 considered equal to 256. The number of B characters affected is either 2T (T ≠ 0) or 512 (T=0).

When command execution is complete, the A field retains its initial contents; the B field's initial contents are replaced by the unpacked data.

If the A and B fields overlap, the results may be different from the results obtained when the two fields do not overlap.

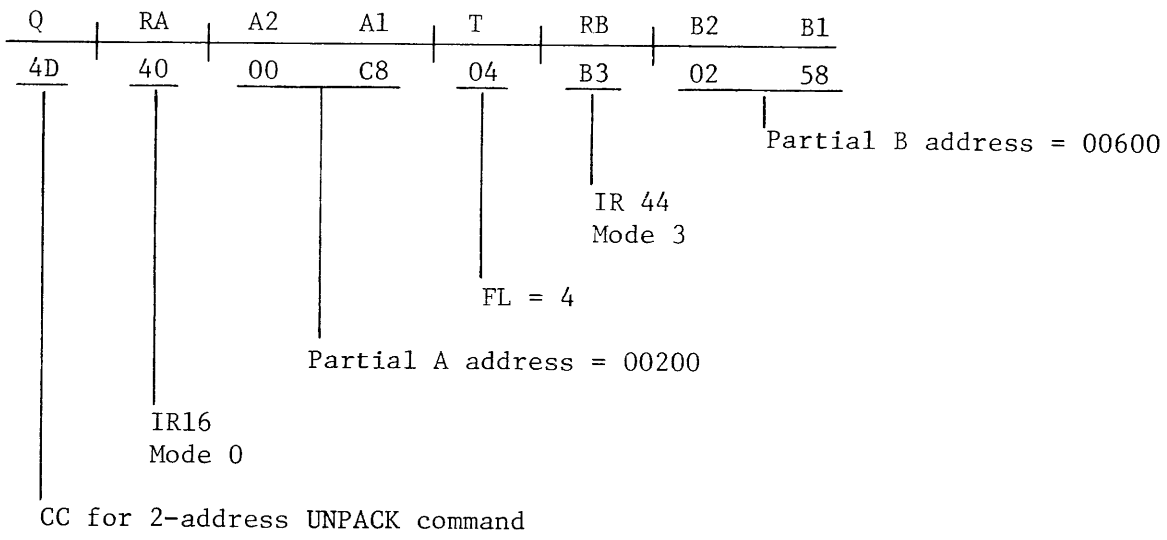
$\underline{B}$ = B + 2T if T is not equal to 0.

$\underline{B}$ = B + 512 if T is equal to 0.

### Command Execution Time

E = 20T + 9

Example 1


Q
RA
A2
A1
T
RB
B2
B1
4D
40
00
C8
04
B3
02
58
Partial B address = 00600
IR 44
Mode 3
FL = 4
Partial A address = 00200
IR16
Mode 0
CC for 2-address UNPACK command

Assume the index registers contain the following:

(IR16) = 0640 (01600)
(IR44) = 1130 (04400)

After command setup:

Effective A address = 0640 + 00C8 = 0708 (01800)
Effective B address = 1130 + 0258 = 1388 (05000)
(IR44) = 1388 (05000) since "Mode 3" (incremental indexing) was specified.

Before command execution:

| A | 0708 | 0709 | 070A | 070B |
|---|---|---|---|---|
| (A) | 0000 1010 | 1010 0010 | 0101 0110 | 0011 1011 |

The contents of B after command setup are not significant.

Following command execution:

| B | 1388 | 1389 | 138A | 138B | 138C |
|---|---|---|---|---|---|
| (B) | 0011 0000 | 0010 1010 | 0010 1010 | 0011 0010 | 0011 0101 |

| 138D | 138E | 138F |
|---|---|---|
| 0011 0110 | 0011 0011 | 0010 1011 |

(A) is unchanged

B = 1388 + 8 = 1390 (05008)

Example 2 (Fields overlap)

| Q | RA | A2 | A1 | T | RB | B2 | B1 |
|---|---|---|---|---|---|---|---|
| 4D | 34 | 01 | A6 | 04 | 00 | 34 | 6D |

Partial B address = 13421

No IR

FL = 4

Partial A address = 00422

IR13
Mode 0

CC for 2-address UNPACK command

Assume the index register contains the following:

(IR13) = 32C8 (13000)

After command setup

Effective A address = 32C8 + 01A6 = 346E (13422)
Effective B address = 346D (13421) - no indexing required

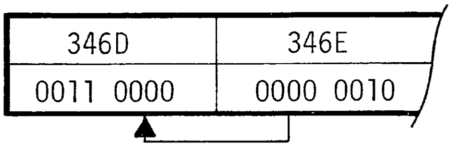
Before command execution

| | OVERLAPPING AREAS | | | |
|---|---|---|---|---|
| 346D | 346E | 346F | 3470 | 3471 |
| 0000 0000 | 0000 0010 | 0111 0100 | 0011 0110 | 0100 1101 |

| 3472 | 3473 | 3474 |
|---|---|---|
| 0000 0000 | 0000 0000 | 0000 0000 |

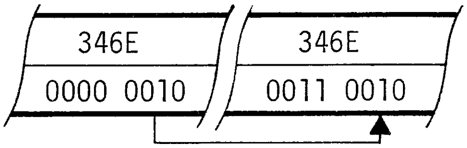
Because the fields overlap, they are altered during command execution as follows:

1. b8 - b5 of 346E replace b4 - b1 of 346D; zone bits 0011 are appended:



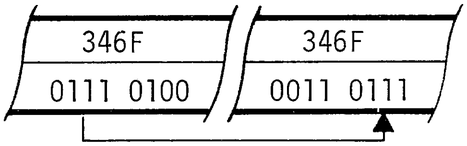
| 346D | 346E |
|---|---|
| 0011 0000 | 0000 0010 |

2. b4 - b1 346E remain unchanged, since, in effect, they replace themselves. Zone bits 0011 replace b8 - b5 of 346E:



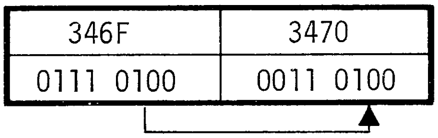
| 346E | 346E |
|---|---|
| 0000 0010 | 0011 0010 |

3. b8 - b5 of 346F replace b4 - b1 of 346F. Zone bits 0011 replace b8 - b5 of 346F:



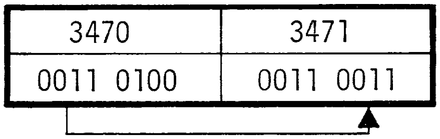
| 346F | 346F |
|---|---|
| 0111 0100 | 0011 0111 |

4. b4 - b1 of 346F replace b4 - b1 of 3470. When the contents of 346F were originally setup for unpacking, all eight bits were moved to a temporary storage area and unpacked from there, so the original b4 - b1 bits of 346F (0100), and not the bits obtained in Step 3 (0111), replace b4 - b1 of 3470:



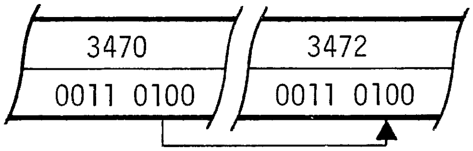
| 346F | 3470 |
|---|---|
| 0111 0100 | 0011 0100 |

5. b8 - b5 of 3470, altered in Step 4, replace b4 - b1 of 3471:



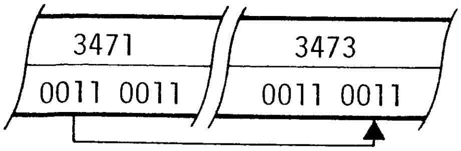
| 3470 | 3471 |
|---|---|
| 0011 0100 | 0011 0011 |

6. b4 - b1 of 3470, altered in Step 4, replace b4 - b1 of 3472:

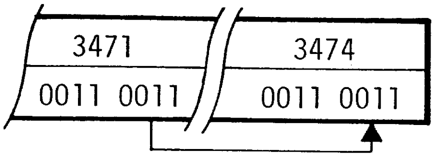


| 3470 | 3472 |
|---|---|
| 0011 0100 | 0011 0100 |

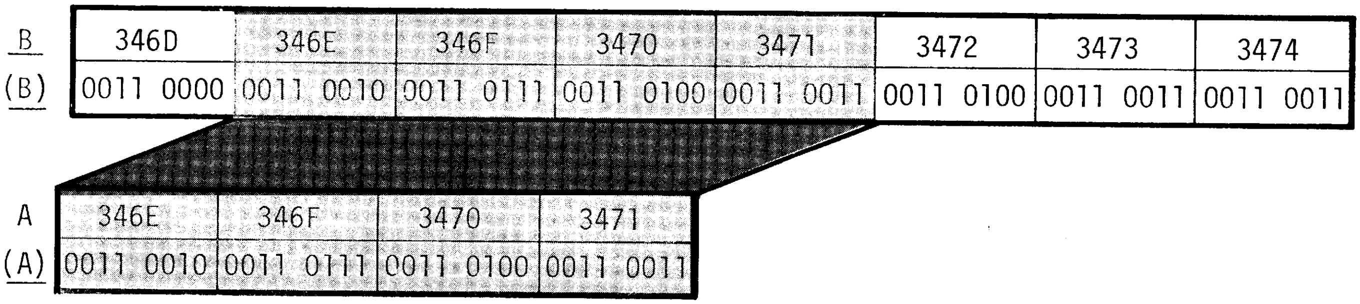
7. b8 - b5 of 3471, altered in Step 5, replace b4 - b1 of 3473:



8. b4 - b1 of 3471, altered in Step 6, replace b4 - b1 of 3474:



Following command execution:



| B | 346D | 346E | 346F | 3470 | 3471 | 3472 | 3473 | 3474 |
|---|---|---|---|---|---|---|---|---|
| (B) | 0011 0000 | 0011 0010 | 0011 0111 | 0011 0100 | 0011 0011 | 0011 0100 | 0011 0011 | 0011 0011 |

| A | 346E | 346F | 3470 | 3471 |
|---|---|---|---|---|
| (A) | 0011 0010 | 0011 0111 | 0011 0100 | 0011 0011 |

B = 346D + 08 = 3475 (13429)

C = 96(60) (EO)

The contents of the field specified by the effective A operand are added binarily to the contents of the field specified by the effective B operand. The result replaces the contents of the B field.

The addition is performed from right to left, eight bits (one byte) at a time. A carry from one byte is added to the next byte to the left; a carry from the leftmost byte is ignored (the overflow flag is not affected). Both fields are considered unsigned and positive.

The T portion of the command designates the number of bytes in each field. T may be any value from 0 through 255, with 0 equivalent to 256.

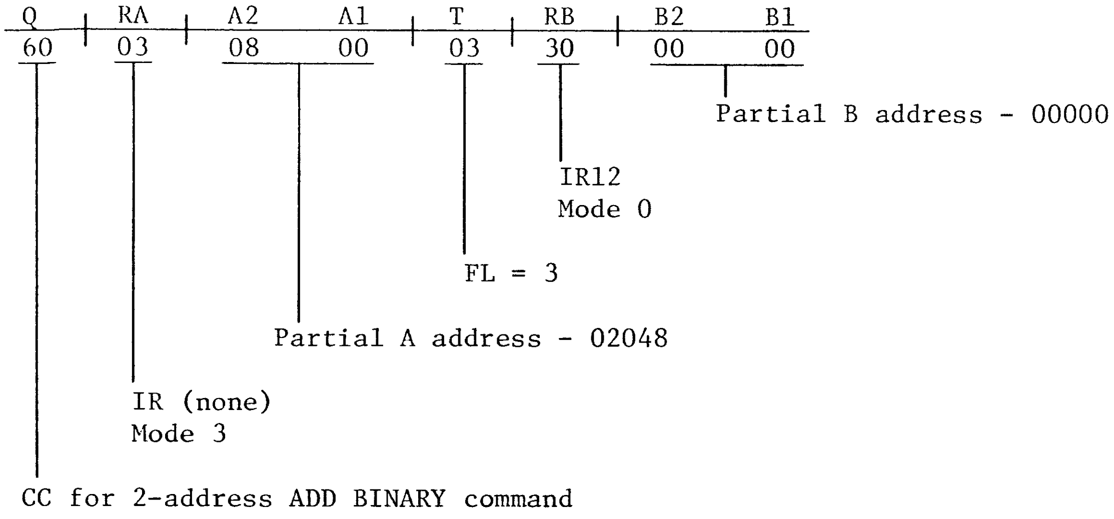
Following command execution, the A field retains its original contents. If the A and B fields overlap, the results may be different from an operation in which the fields do not overlap. In this case, the initial contents of the A field are altered.

$\underline{B} = B$

Command Execution Time

E = 9 + 9T

Example 1.



| Q | RA | A2 | A1 | T | RB | B2 | B1 |
|---|---|---|---|---|---|---|---|
| 60 | 03 | 08 | 00 | 03 | 30 | 00 | 00 |

Partial B address - 00000
IR12
Mode 0
FL = 3
Partial A address - 02048
IR (none)
Mode 3
CC for 2-address ADD BINARY command

Assume IR12 contains 04B0 (01200)

After command setup

Effective A address = 0800 (02048). Since no indexing is required, the specification of Mode 3 is meaningless in this case.

Effective B address = 04B0 + 0000 = 04B0 (01200)

Before command execution:

| A | 0800 | 0801 | 0802 |
|---|---|---|---|
| (A) | 1010 1100 | 1100 1010 | 0010 1101 |

| B | 04B0 | 04B1 | 04B2 |
|---|---|---|---|
| (B) | 1110 0110 | 1011 1110 | 0111 0101 |

Addition (For a more detailed explanation, refer to the Binary Arithmetic section of the NCR CENTURY 100 PROCESSOR manual).

```
(A)  10101100   11001010   00101101
(B)  11100110   10111110   01110101
     ------------------------------
     01001010   01110100   01011000   Partial Sum
   1  1  1  1       1 1     1  1 1    Carries
   --------------------------------
   1 10010011   10001000   10100010   Final Sum
    The carry beyond the specified field length is ignored.
```

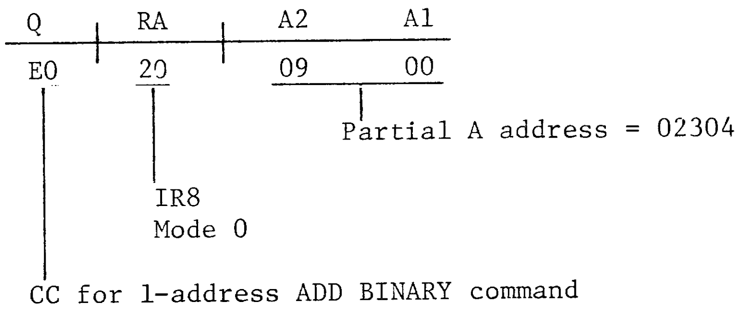
Following command execution:

(A) = Unchanged

| B | 04B0 | 04B1 | 04B2 |
|---|---|---|---|
| (B) | 1001 0011 | 1000 1000 | 1010 0010 |

B = 04B0 (01200)

Example 2. (Fields overlap)



Partial A address = 02304

IR8
Mode 0

CC for 1-address ADD BINARY command

Assume IR8 contains FFFD (65,533). Since indexing on the NCR Century 100 is cyclic:

Effective A address = FFFD + 0900 = 08FD (02301).

Since a 1-address command is specified, the B and T values stored from a previous command are used. Assume that the T and B registers contain 04 and 08FC (02300), respectively.

Before command execution:

| A | 08FD | 08FE | 08FF | 0900 |
|---|---|---|---|---|
| (A) | 0001 0100 | 0010 1011 | 0011 1101 | 1111 0010 |

| B | 08FC | 08FD | 08FE | 08FF |
|---|---|---|---|---|
| (B) | 0000 0000 | 0001 0100 | 0010 1011 | 0011 1101 |

Because the fields overlap, addition takes place as follows:

1. The contents of 0900 are added to the contents of 08FF, and the result is stored in 08FF.

```
  11110010
  00111101
  11001111   Partial Sum
   11        Carries
1 00101111   Final Sum
 ↖Carry to next byte
```

2. The contents of 08FF (altered by Step 1) are added to the contents of 08FE, and the result is stored in 08FE.

```
00101111
00101011
00000100   Partial Sum
 1 1 111   Carries (in-
01011011   cluding initial
           carry from
           Step 1).
```

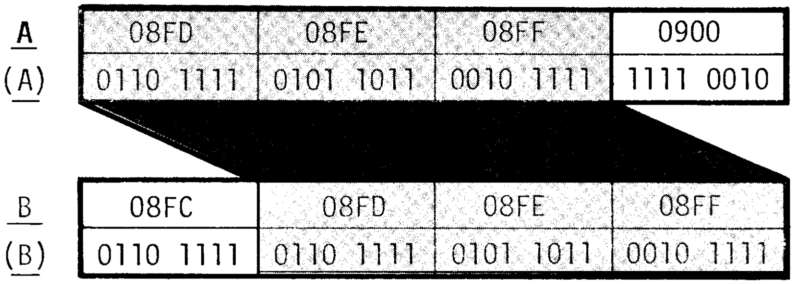
3. The contents of 08FE (altered by Step 2) are added to the contents of 08FD, and the result is stored in 08FD.

```
01011011
00010100
01001111   Partial Sum
   1       Carry
01101111   Final Sum
```

4. The contents of 08FD (altered by Step 3) are added to the contents of 08FC, and the result is stored in 08FC.

```
01101111
00000000
01101111   Partial & Final
           Sum
```

Following Command execution:



| A | 08FD | 08FE | 08FF | 0900 |
|---|---|---|---|---|
| (A) | 0110 1111 | 0101 1011 | 0010 1111 | 1111 0010 |

| B | 08FC | 08FD | 08FE | 08FF |
|---|---|---|---|---|
| (B) | 0110 1111 | 0110 1111 | 0101 1011 | 0010 1111 |

B = 08FC (02300)

C = 97(61)(E1)

The contents of the field specified by the effective A operand are subtracted binarily from the contents of the field specified by the effective B operand; the results replace the initial contents of the B field.

Subtraction is performed from right to left, one character (byte) at a time. As explained in the NCR CENTURY 100 PROCESSOR manual, subtraction is actually performed by complementary addition. The 1's complement of the A character is formed and added to the B character to derive a partial sum. Carries from the first addition and an initial carry (to compensate for use of the 1's complement) are added to the partial sum to derive the final result. A carry beyond the leftmost character position in the specified field is ignored (the overflow flag is not affected). Both A and B fields are considered unsigned and positive.

The T portion of the command specifies the number of characters in each field; T may specify any number from 0 through 255, with 0 equivalent to 256.

At completion of the command, the A field retains its initial contents. The initial contents of the B field are replaced by the results of the subtraction. If the contents of the A field are greater than the contents of the B field, the final result stored is the 2's complement of (A) - (B).

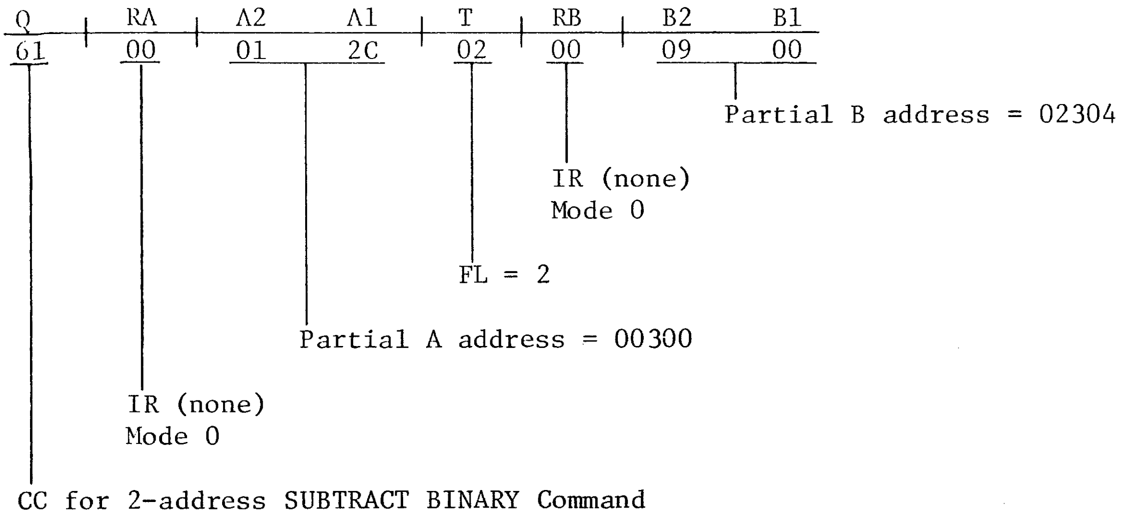
If the A and B fields overlap, the results may be different from the operation in which the fields do not overlap. The initial contents of the A field will also change.

B = B

Command Execution Time

E = 9 + 9T

Example 1.


Q
RA
A2
A1
T
RB
B2
B1
61
00
01
2C
02
00
09
00
Partial B address = 02304
IR (none)
Mode 0
FL = 2
Partial A address = 00300
IR (none)
Mode 0
CC for 2-address SUBTRACT BINARY Command

After command setup:

Effective A address = 012C (00300)
Effective B address = 0900 (02304)

Before command execution:

| | | |
|---|---|---|
| A | 012C | 012D |
| (A) | 0000 0000 | 0010 1000 |

| | | |
|---|---|---|
| B | 0900 | 0901 |
| (B) | 0001 0010 | 0110 1100 |

Subtraction:

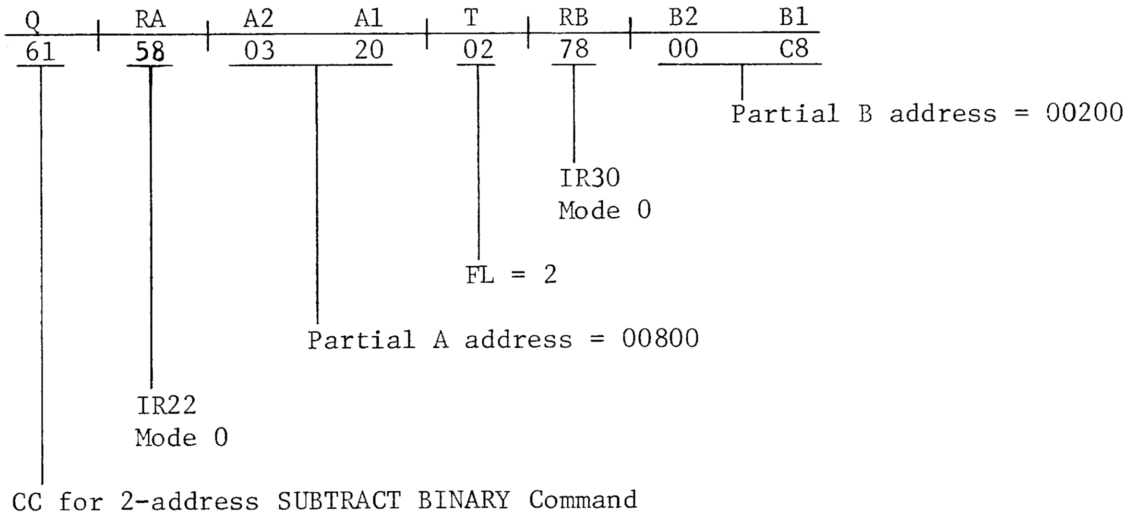
```
   11111111    11010111    1's complement of (A)
   00010010    01101100
   ____________________
   11101101    10111011    Partial Sum
   11111111    1  1        Carries
                     1     Initial Carry to form 2's complement
   ____________________
 1 00010010    01000100
 ↖
  Carry beyond the specified field length is ignored.
```

After command execution:

(A) - unchanged

| B | 0900 | 0901 |
|---|---|---|
| (B) | 0101 0010 | 0100 0100 |

Example 2. (Contents of effective A address > contents of effective B address)


Q
61
RA
58
A2
03
A1
20
T
02
RB
78
B2
00
B1
C8
Partial B address = 00200
IR30
Mode 0
FL = 2
Partial A address = 00800
IR22
Mode 0
CC for 2-address SUBTRACT BINARY Command

Assume that the index registers contain the following:

(IR22) = 0898 (02200)
(IR30) = 0BB8 (03000)

After command setup:

Effective A address = 0898 + 0320 = 0BB8 (03000)
Effective B address = 0BB8 + 00C8 = 0C80 (03200)

Before command execution:

| A | 0BB8 | 0BB9 |
|---|---|---|
| (A) | 0111 0101 | 1010 0100 |

| B | 0C80 | 0C81 |
|---|---|---|
| (B) | 0100 0011 | 0110 1101 |

Subtraction:

```
 10001010     01011011      1's complement of A
 01000011     01101101
 _________________________
 11001001     00110110      Partial Sum
        1     1  1  1       Carries
                     1      Initial Carry
 _________________________
 11001101     11001001
```

Following command execution:

(A) - unchanged

|  |  |  |
|---|---|---|
| B | 0C80 | 0C81 |
| (B) | 1100 1101 | 1100 1001 |

C = 98(62)(E2)

The contents of the field specified by the effective A address are added decimally to the contents of the field specified by the effective B address (see addition tables, page 24). The result of the addition replaces the contents of the B field.

Decimal (BCD) addition is performed from right to left, one byte at a time. Only b4 - b1 of each byte are added. The four most-significant bits of the byte are ignored, but the configuration 0011 is stored in b8 - b5 of each byte in the result.

Effectively, each A field character is added to excess six (see Binary Arithmetic section of the NCR CENTURY 100 PROCESSOR manual), and then added to the corresponding B field character. If the addition of the B field character causes a carry beyond the four bits, the result is stored in b4 - b1 of the B field character and the carry is added to the next character position to the left. If no carry occurs, logic circuitry corrects for the excess six and stores the corrected four bits in b4 - b1 of the corresponding B field character. A carry beyond the leftmost position of the defined field is discarded, but the overflow flag (OF) is turned ON.

The T portion of the command designates the number of characters in each operand field; T may be any value from 0 through 255, with 0 equivalent to 256.

On completion of the ADD UNSIGNED command, the A field retains its initial contents; the initial contents of the B field are replaced by (A) + (B).

If the A and B fields overlap, the results may be different from the results obtained where the fields do not overlap. In this case, the initial contents of the A field are changed.

Packed data must be unpacked before execution of this command.

$\underline{B} = B$

Command Execution Time

E = 9 + 9T if no overflow<br>
E = 11 + 9T if overflow

## ADDITION TABLES

### NO CARRY FROM PREVIOUS DIGIT-POSITION
DIGITS FROM (A)

DIGITS FROM (B)

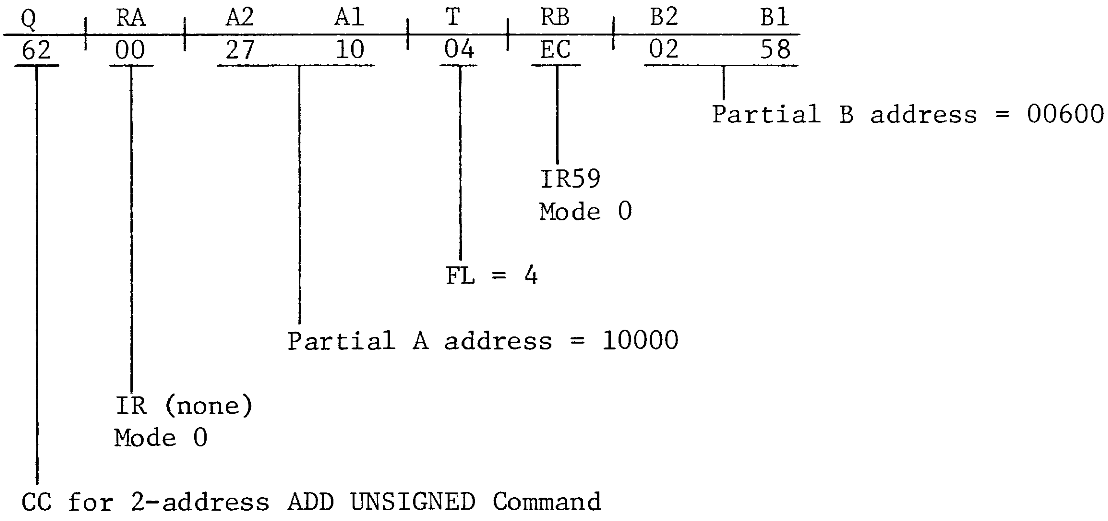
| (B) \ (A) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | B | C | D | E | F | 0 |
| 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | C | D | E | F | 0 | 1 |
| 3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | D | E | F | 0 | 1 | 2 |
| 4 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | E | F | 0 | 1 | 2 | 3 |
| 5 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | F | 0 | 1 | 2 | 3 | 4 |
| 6 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | 0 | 1 | 2 | 3 | 4 | 5 |
| 7 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | 1 | 2 | 3 | 4 | 5 | 6 |
| 8 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | 2 | 3 | 4 | 5 | 6 | 7 |
| 9 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | 3 | 4 | 5 | 6 | 7 | 8 |
| A | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | 4 | 5 | 6 | 7 | 8 | 9 |
| B | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | cA | 5 | 6 | 7 | 8 | 9 | c0 |
| C | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | cA | cB | 6 | 7 | 8 | 9 | c0 | c1 |
| D | c3 | c4 | c5 | c6 | c7 | c8 | c9 | cA | cB | cC | 7 | 8 | 9 | c0 | c1 | c2 |
| E | c4 | c5 | c6 | c7 | c8 | c9 | cA | cB | cC | cD | 8 | 9 | c0 | c1 | c2 | c3 |
| F | c5 | c6 | c7 | c8 | c9 | cA | cB | cC | cD | cE | 9 | c0 | c1 | c2 | c3 | c4 |

### CARRY FROM PREVIOUS DIGIT-POSITION
DIGITS FROM (A)

DIGITS FROM (B)

| (B) \ (A) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | B | C | D | E | F | 0 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | C | D | E | F | 0 | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | D | E | F | 0 | 1 | 2 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | E | F | 0 | 1 | 2 | 3 |
| 4 | 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | F | 0 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | 0 | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | 2 | 3 | 4 | 5 | 6 | 7 |
| 8 | 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | 4 | 5 | 6 | 7 | 8 | 9 |
| A | c1 | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | cA | 5 | 6 | 7 | 8 | 9 | c0 |
| B | c2 | c3 | c4 | c5 | c6 | c7 | c8 | c9 | cA | cB | 6 | 7 | 8 | 9 | c0 | c1 |
| C | c3 | c4 | c5 | c6 | c7 | c8 | c9 | cA | cB | cC | 7 | 8 | 9 | c0 | c1 | c2 |
| D | c4 | c5 | c6 | c7 | c8 | c9 | cA | cB | cC | cD | 8 | 9 | c0 | c1 | c2 | c3 |
| E | c5 | c6 | c7 | c8 | c9 | cA | cB | cC | cD | cE | 9 | c0 | c1 | c2 | c3 | c4 |
| F | c6 | c7 | c8 | c9 | cA | cB | cC | cD | cE | cF | c0 | c1 | c2 | c3 | c4 | c5 |

Example 1.



Partial B address = 00600

IR59
Mode 0

FL = 4

Partial A address = 10000

IR (none)
Mode 0

CC for 2-address ADD UNSIGNED Command

Assume IR59 contains 170C (05900)

After command setup:

Effective A address = 2710 (10000) - no indexing required
Effective B address = 170C + 0258 = 1964 (06500)

Before command execution:

| | | | | |
|---|---|---|---|---|
| A | 2710 | 2711 | 2712 | 2713 |
| (A) | 0011 0100 | 0011 0111 | 0011 0010 | 0011 1000 |

| | | | | |
|---|---|---|---|---|
| B | 1964 | 1965 | 1966 | 1967 |
| (B) | 0011 0110 | 0011 0100 | 0011 0100 | 0011 0011 |

## Addition:

Decimal equivalent of addition to be performed:

```
   4728
 + 6443
 ------
  11171
```

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

```
    0100  0111  0010  1000
    0110  0110  0110  0110
    ----------------------
    0010  0001  0100  1110    Partial Sum
    1     11     1            Carries
    ----------------------
    1010  1101  1000  1110    Final Sum
```

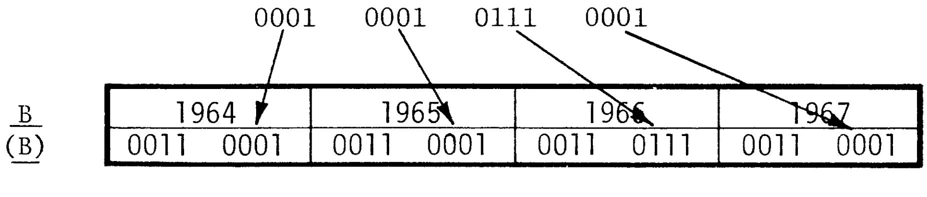
2. B4 - b1 of each corresponding B field character are added to the result obtained in Step 1:

```
    1010  1101  1000  1110
    0110  0100  0100  0011
    ----------------------
    1100  1001  1100  1101    Partial Sum
     1    1            1      Carries
    ----------------------
  1 0001  0001  1101  0001    Uncorrected Final Sum
  ↖
   Carry sets OF ON
```

3. Since 1101 is not a valid BCD number (its binary value > 9), hardware makes a decimal correction by, in effect, adding the 2's complement of 6:

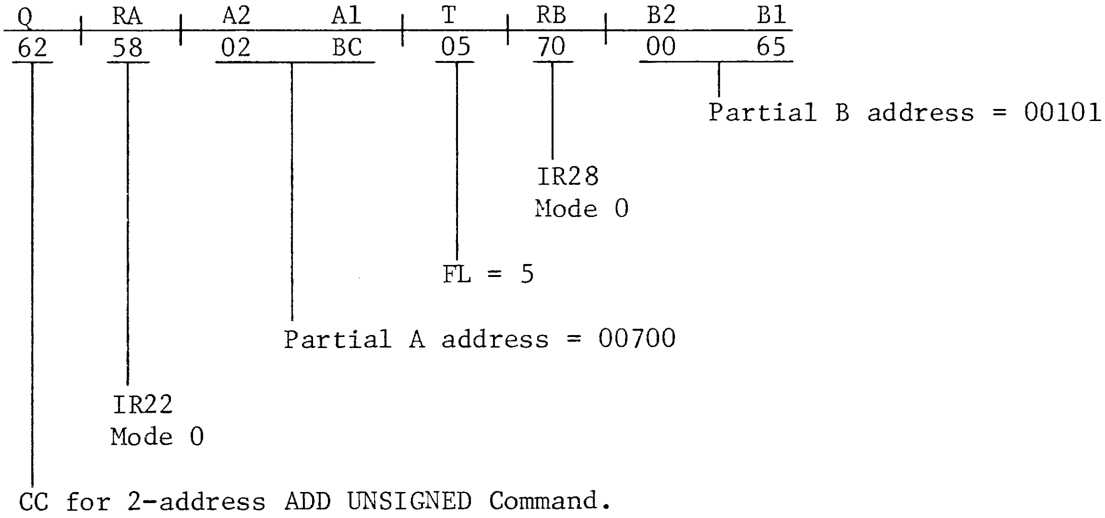
```
    1101
    1010
    ----
  1 0111
  ↖
   Carry is ignored
```

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



Decimal value in NCR Century code = 1171, with OF ON, indicating overflow carry of 1.

Example 2. (Fields overlap)



| Q | RA | A2 | A1 | T | RB | B2 | B1 |
|---|---|---|---|---|---|---|---|
| 62 | 58 | 02 | BC | 05 | 70 | 00 | 65 |

Partial B address = 00101

IR28
Mode 0

FL = 5

Partial A address = 00700

IR22
Mode 0

CC for 2-address ADD UNSIGNED Command.

Assume the index registers contain the following:

(IR22) = 0898 (02200)
(IR28) = OAFO (02800)

After command setup

Effective A address = 0898 + 02BC = OB54 (02900)
Effective B address = OAFO + 0065 = OB55 (02901)

Before command execution

| OB54 | OB55 | OB56 | OB57 | OB58 | OB59 |
|---|---|---|---|---|---|
| 0011 0100 | 0011 0001 | 0011 0010 | 0011 0110 | 0011 1001 | 0011 0000 |

Overlap Area

Addition:

Decimal equivalent of addition to be performed:

```
  41269
+ 12690
  53959
```

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

```
0100    0001    0010    0110    1001
0110    0110    0110    0110    0110
------------------------------------
0010    0111    0100    0000    1111     Partial Sum
1               1       11               Carries
------------------------------------
1010    0111    1000    1100    1111     Final Sum
```

2. b4 - b1 of each corresponding B field character are added to the results obtained in Step 1:

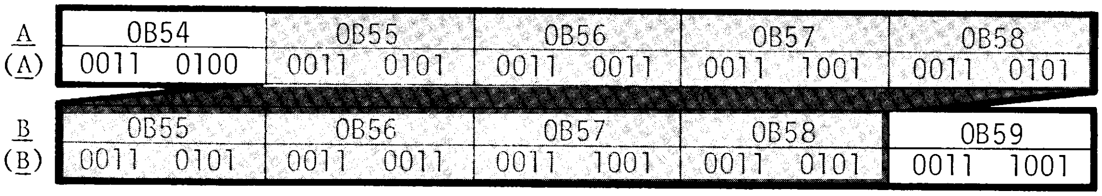
```
1010    0111    1000    1100    1111
0001    0010    0110    1001    0000
------------------------------------
1011    0101    1110    0101    1111     Partial Sum
           1       1                     Carries
------------------------------------
1011    1001    1111    0101    1111     Final Sum
```

3. The sums of those additions which resulted in an illegal BCD number (binary value > 9) or which did not generate a carry beyond b4 are decimally corrected by, in effect, subtracting the excess six:

```
1011    1001    1111    0101    1111
0110    0110    0110            0110
------------------------------------
0101    0011    1001    0101    1001
```

4. The corrected results replace b4 - b1 of each corresponding B character. Note that, in this case, the overlap did not cause the sum to be any different from the sum that would be obtained if the fields did not overlap. Only (A) is affected.

Following command execution:



| | | | | | |
|---|---|---|---|---|---|
| A | 0B54 | 0B55 | 0B56 | 0B57 | 0B58 |
| (A) | 0011 0100 | 0011 0101 | 0011 0011 | 0011 1001 | 0011 0101 |
| B | 0B55 | 0B56 | 0B57 | 0B58 | 0B59 |
| (B) | 0011 0101 | 0011 0011 | 0011 1001 | 0011 0101 | 0011 1001 |

BCD Value = 53959

C = 99(63)(E3)

The contents of the field specified by the effective A address are subtracted decimally from the contents of the field specified by the effective B address (see subtraction tables, page 30). The result of the subtraction replaces the contents of the B field.

Decimal (BCD) subtraction is performed from right to left, one byte at a time. Only b4 - b1 of each byte are used. The four most-significant bits of each byte are ignored, but the zone-bit configuration 0011 is stored in b8 - b5 of each byte in the result.

The 1's complement of the A-bit configuration is formed and added to the B-bit configuration to derive a partial sum. Carries from the first addition and an initial carry (to compensate for the use of the 1's complement) are added to the partial sum to derive the final sum. A carry beyond the leftmost bit position is not included in the result, but is used by the processor to determine whether the result was positive (contents of B > contents of A) or negative (contents of A > contents of B). If there is a carry beyond the leftmost bit position, the result is positive and is stored in b4 - b1 of the corresponding B character. No carry indicates that the result is negative and the processor makes a decimal correction (in effect, subtracting excess six from the result) and stores the 10's complement of (A) - (B).

The T portion of the command specifies the number of characters in each field; T may be any number from 0 through 255, with 0 equivalent to 256.

At completion of command execution, the A field retains its initial contents. The initial contents of the B field are replaced by the result of the subtraction.

If the A and B fields overlap, the results may be different from the results of an operation where the fields do not overlap. The initial contents of the A field will also be altered.

Packed fields must be unpacked before executing this command.

$\underline{B}$ = B

Command Execution Time

E = 9 + 9T if no overflow occurs
E = 11 + 9T if overflow occurs

## SUBTRACTION TABLES

**NO BORROW BY PREVIOUS DIGIT-POSITION**
**DIGITS FROM (A)**

DIGITS FROM (B)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 | $^B$0 | $^B$F | $^B$E | $^B$D | $^B$C | $^B$B |
| **1** | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 | $^B$0 | $^B$F | $^B$E | $^B$D | $^B$C |
| **2** | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 | $^B$0 | $^B$F | $^B$E | $^B$D |
| **3** | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 | $^B$0 | $^B$F | $^B$E |
| **4** | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 | $^B$0 | $^B$F |
| **5** | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 | $^B$0 |
| **6** | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 |
| **7** | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 |
| **8** | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 |
| **9** | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 |
| **A** | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 |
| **B** | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 |
| **C** | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 |
| **D** | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 |
| **E** | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 |
| **F** | F | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

(A)ı

(B)ı

**BORROW BY PREVIOUS DIGIT-POSITION**
**DIGITS FROM (A)**

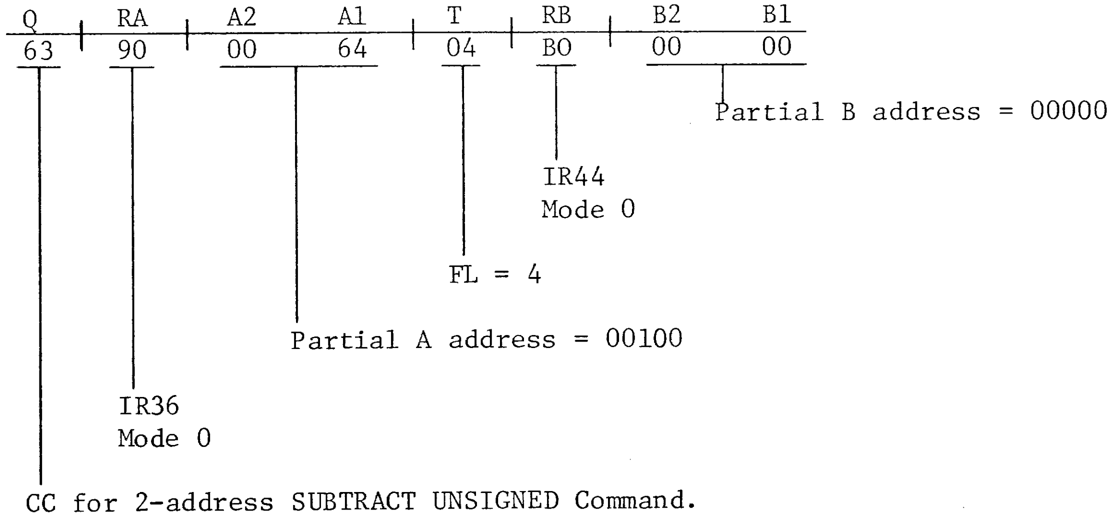
DIGITS FROM (B)

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 | $^B$0 | $^B$F | $^B$E | $^B$D | $^B$C | $^B$B | $^B$A |
| **1** | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 | $^B$0 | $^B$F | $^B$E | $^B$D | $^B$C | $^B$B |
| **2** | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 | $^B$0 | $^B$F | $^B$E | $^B$D | $^B$C |
| **3** | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 | $^B$0 | $^B$F | $^B$E | $^B$D |
| **4** | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 | $^B$0 | $^B$F | $^B$E |
| **5** | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 | $^B$0 | $^B$F |
| **6** | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 | $^B$0 |
| **7** | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 | $^B$1 |
| **8** | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 | $^B$2 |
| **9** | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 | $^B$3 |
| **A** | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 | $^B$4 |
| **B** | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 | $^B$5 |
| **C** | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 | $^B$6 |
| **D** | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 | $^B$7 |
| **E** | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 | $^B$8 |
| **F** | E | D | C | B | A | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $^B$9 |

(A)ı

(B)ı

Example 1.


Q
63
RA
90
A2
00
A1
64
T
04
RB
B0
B2
00
B1
00
Partial B address = 00000
IR44
Mode 0
FL = 4
Partial A address = 00100
IR36
Mode 0
CC for 2-address SUBTRACT UNSIGNED Command.

Assume the index registers contain the following:

(IR36) = 0E10 (03600)
(IR44) = 1130 (04400)

After command setup:

Effective A address = 0E10 + 0064 = 0E74 (03700)
Effective B address = 1130 + 0000 = 1130 (04400)

Before command execution:

| | | | | |
|---|---|---|---|---|
| A | 0E74 | 0E75 | 0E76 | 0E77 |
| (A) | 0011 0100 | 0011 0100 | 0011 0010 | 0011 1000 |

| | | | | |
|---|---|---|---|---|
| B | 1130 | 1131 | 1132 | 1133 |
| (B) | 0011 0111 | 0011 0010 | 0011 0011 | 0011 1001 |

## Subtraction

Decimal equivalent of the subtraction to be performed:

```
  7239
- 4428
  2811
```

1. As each A-field configuration is read in, it is converted to its 1's complement:

| A-field Bits | 1's Complement |
|---|---|
| 0100 | 1011 |
| 0100 | 1011 |
| 0010 | 1101 |
| 1000 | 0111 |

2. The complements are then added to the corresponding B characters:

```
   1011    1011    1101    0111
   0111    0010    0011    1001
   1100    1001    1110    1110      Partial Sum
    11      1       1       1        Carries
      1       1       1       1      Initial Carries
 c 0010   c1110   c0001   c0001      Uncorrected Result

   This carry not stored
```

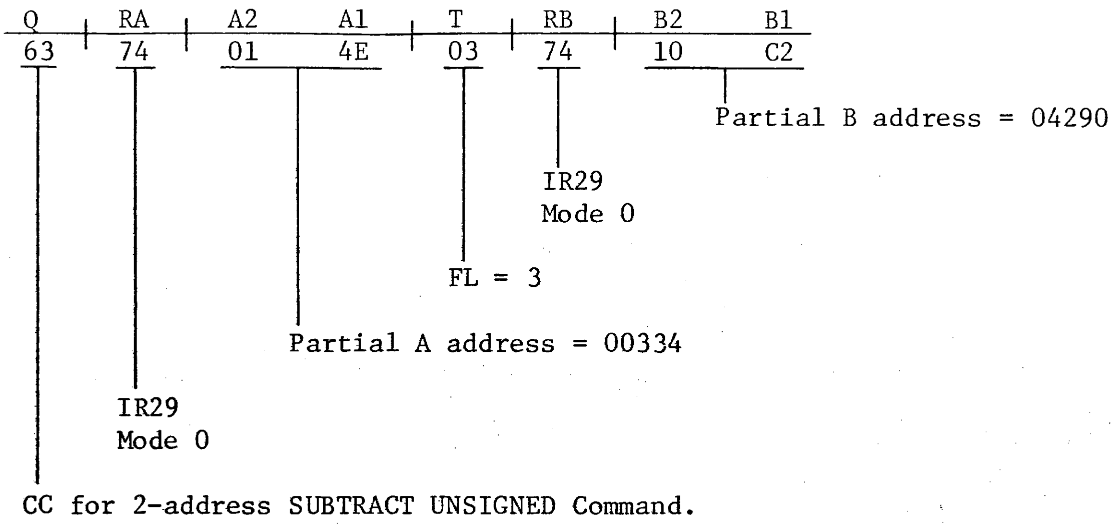
3. Since there was no carry from third set of bits added, their sum must be corrected:

```
   0010    1110    0001    0001      Uncorrected Result
           0110                      Excess Six Correction
   0010    1000    0001    0001      Corrected Result
```

4. The results are then stored in b4 - b1 of the corresponding B-field characters, and b8 - b5 of each character are set to 0011:

| B | 1130 | 1131 | 1132 | 1133 | |
|---|---|---|---|---|---|
| (B) | 0011 0010 | 0011 1000 | 0011 0001 | 0011 0001 | BCD Value = 2811 |

Example 2. (A) > (B)



Partial B address = 04290

IR29
Mode 0

FL = 3

Partial A address = 00334

IR29
Mode 0

CC for 2-address SUBTRACT UNSIGNED Command.

Assume IR29 contains 067E (01662)

After command setup:

Effective A address = 067E + 014E = 07CC (01996)
Effective B address = 067E + 10C2 = 1740 (05952)

Before command execution

| | | | |
|---|---|---|---|
| A | 07CC | 07CD | 07CE |
| (A) | 0011 0101 | 0011 0100 | 0011 0010 |

| | | | |
|---|---|---|---|
| B | 1740 | 1741 | 1742 |
| (B) | 0011 0100 | 0011 0110 | 0011 0100 |

## Subtraction

Decimal equivalent of subtraction to be performed

```
  464
- 542
- 078
```

Since the result was negative, the 10's complement of (A) - (B) is stored:

```
  542       1000
- 464     -  078
  078        922     10's complement of 078
```

1. The 1's complement is formed for each set of A-field bits:

| A-field Bits | 1's Complement |
|---|---|
| 0101 | 1010 |
| 0100 | 1011 |
| 0010 | 1101 |

2. The complemented A-field bits are added to the corresponding B-field bits:

| | | | |
|---|---|---|---|
| 1010 | 1011 | 1101 | |
| 0100 | 0110 | 0100 | |
| 1110 | 1101 | 1001 | Partial sum |
| | 1 | 1 | Carries |
| 1 | 1 | 1 | Initial Carries |
| 1111 | $^{c}$0010 | $^{c}$0010 | Uncorrected Result |

3. Since there was no carry beyond b4 of the leftmost digit position, the four leftmost bits must be corrected:

| | | | |
|---|---|---|---|
| 1111 | 0010 | 0010 | Uncorrected Result |
| 0110 | | | Excess Six |
| 1001 | 0010 | 0010 | Corrected Result |

4. The corrected results are stored in b4 - b1 of the corresponding B-field characters, and b8 - b5 of each character are set to 0011:

| B | 1740 | 1741 | 1742 |
|---|---|---|---|
| (B) | 0011 1001 | 0011 0010 | 0011 0010 |

BCD Value = 922 = 10's complement of (A) - (B)

C = 100(64)(E4)

The contents of the field specified by the effective A address are moved into the field specified by the effective B address. Moving is done from right to left, one character at a time, with each A-field character replacing the corresponding B-field character.

The T portion of the command specifies the number of characters to be moved, and may range in value from 0 through 255, with 0 equivalent to 256.

At the completion of the command, the A field retains its initial contents. If the A and B fields overlap, the results may be different from the operation where the fields do not overlap. In this case, the initial contents of the A field are always changed.

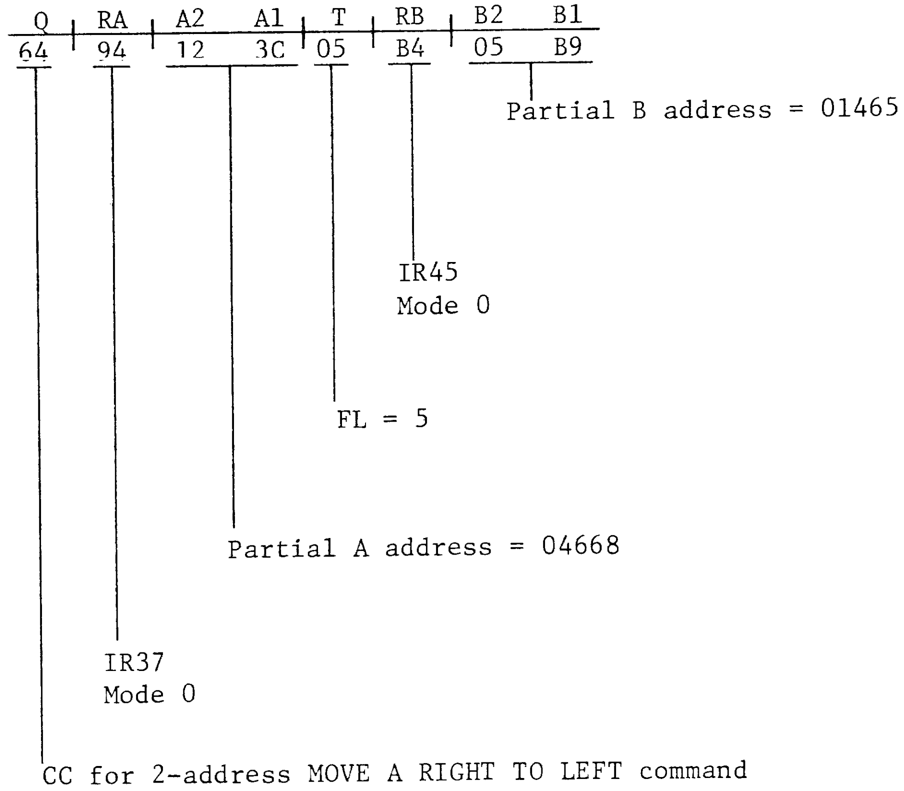
B = B

Command Execution Time

E = 9 + 8T

EXAMPLES

Example 1.


Q
RA
A2
A1
T
RB
B2
B1
64
94
12
3C
05
B4
05
B9
Partial B address = 01465
IR45
Mode 0
FL = 5
Partial A address = 04668
IR37
Mode 0
CC for 2-address MOVE A RIGHT TO LEFT command

Assume the index registers contain the following:

(IR37) = OE74 (03700)
(IR45) = 1194 (04500)

After command setup:

Effective A address = OE74 + 123C = 20BO (08368)
Effective B address = 1194 + 05B9 = 174D (05965)

Before command execution:

| A | 20B0 | 20B1 | 20B2 | 20B3 | 20B4 |
|---|---|---|---|---|---|
| (A) | 0100 0011 | 0010 1101 | 0011 0110 | 0011 0001 | 0011 0101 |

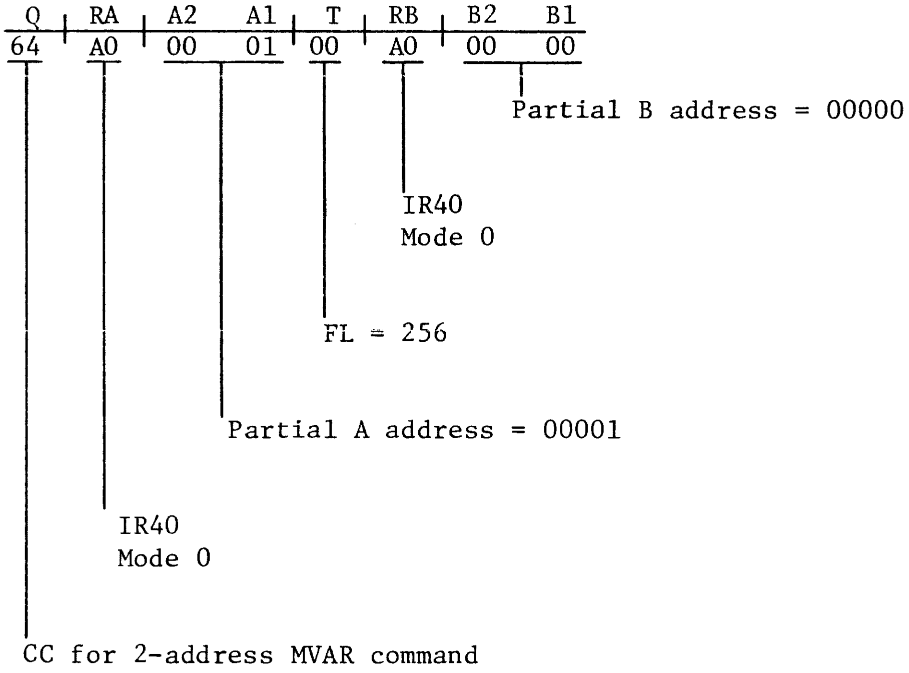
The contents of B are not significant.

Following command execution:

$(\underline{A}) = (A)$

| B | 174D | 174E | 174F | 1750 | 1751 |
|---|---|---|---|---|---|
| $(\underline{B})$ | 0100 0011 | 0010 1101 | 0011 0110 | 0011 0001 | 0011 0101 |

Example 2. (Fields overlap)


Q
64
RA
A0
A2
00
A1
01
T
00
RB
A0
B2
00
B1
00
Partial B address = 00000
IR40
Mode 0
FL = 256
Partial A address = 00001
IR40
Mode 0
CC for 2-address MVAR command

Assume index register 40 contains 1000 (04096).

After command setup:

Effective A address = 1000 + 0001 = 1001 (04097)
Effective B address = 1000 + 0000 = 1000 (04096)

Before command execution:

| | | | | | |
|---|---|---|---|---|---|
| A | 1001 | 1002 | 10EF | 10FF | 1100 |
| (A) | 1111 1011 | 0010 1100 | 0011 0010 | 1100 0100 | 0000 0000 |

Locations 1003 through 10DF

The contents of B are not significant.

When the move is initiated, the contents of 1100 replace the contents of 10EF. Because of the overlap, on each successive move the character 00000000 will replace the corresponding B-field character.

Following command execution:

| | | | | | |
|---|---|---|---|---|---|
| A | 1001 | 1002 | 10EF | 10FF | 1100 |
| (A) | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 |

| | | | | |
|---|---|---|---|---|
| B | 1000 | 1001 | 10EF | 10FF |
| (B) | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 |

C = 101(65)(E5)

The contents of the field specified by the Effective A address are compared binarily to the contents of the field specified by the effective B address, and a G(reater), L(ess), or E(qual) flag is turned ON to indicate the result of the comparison.

The binary comparison is performed one character at a time from right to left. Effectively, the complement of the A field character is added to the B field character, as in a binary subtraction operation. If the contents of A are less than the contents of B, the addition results in a carry beyond the leftmost character in the field, and the number unequal indicator is turned ON. This combination causes the L flag to be set ON.

If the contents of A and B are equal, the addition also results in a carry beyond the leftmost character in the field, but the number unequal indicator is turned OFF, or remains OFF. This combination sets the E flag ON.

If the contents of A are greater than the contents of B, no carry occurs beyond the leftmost character position. This causes the G flag to be set ON.

The conditions (A)>(B) and (A) = (B) also cause the repeat indicator (RI) to be turned OFF. The condition (A)<(B) does not affect the status of the RI.

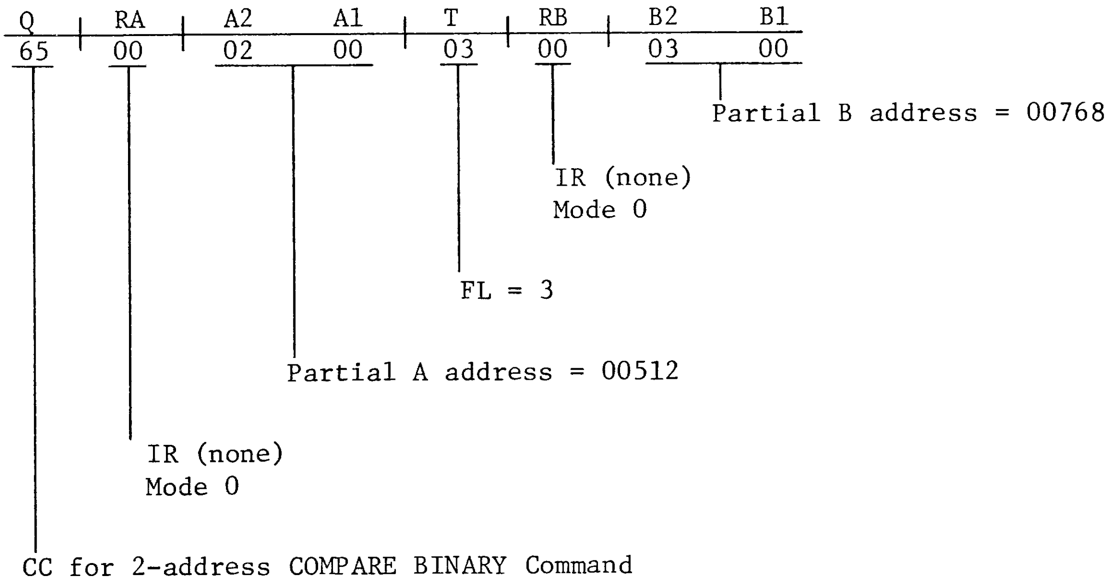
T may range in value from 0 through 255, with 0 equivalent to 256.

$\underline{B}$ = B

Command Execution Time

E = 11 + 9T

Example 1. (A) < (B)


Q
RA
A2
A1
T
RB
B2
B1
65
00
02
00
03
00
03
00
Partial B address = 00768
IR (none)
Mode 0
FL = 3
Partial A address = 00512
IR (none)
Mode 0
CC for 2-address COMPARE BINARY Command

After command setup:

Effective A address = 0200 (00512) - No indexing required.
Effective B address = 0300 (00768) - No indexing required.

Before command execution

| | | | |
|---|---|---|---|
| A | 0200 | 0201 | 0202 |
| (A) | 0011 0111 | 0011 0110 | 0011 1001 |

| | | | |
|---|---|---|---|
| B | 0300 | 0301 | 0302 |
| (B) | 0011 1001 | 0011 0100 | 0011 1001 |

Binary Comparison:

1. The 1's complement is derived for each A-field character:

| A-field Character | 1's Complement of A-field Character |
|---|---|
| 00110111 | 11001000 |
| 00110110 | 11001001 |
| 00111001 | 11000110 |

2. The 1's complements of the A-field characters are added to the corresponding B-field characters:

```
           11001000   11001001   11000110
           00111001   00110100   00111001
           ------------------------------
           11110001   11111101   11111111    Partial Sum
              1                              Carry
                                        1    Initial Carry
           ------------------------------
         1 00000001   11111110   00000000    Final Sum
```

3. Because the result is not equal to zero, the number unequal indicator is turned ON.

4. Because there is a carry beyond the leftmost character in the specified field, and the number unequal indicator is ON, the L flag is set ON.

Example 2. (A) = (B)

Assume that the fields defined in Example 1 had contained the following:

|  |  |  |  |
|---|---|---|---|
| A | 0200 | 0201 | 0202 |
| (A) | 0100 0001 | 0011 1101 | 0100 0010 |

|  |  |  |  |
|---|---|---|---|
| B | 0300 | 0301 | 0302 |
| (B) | 0100 0001 | 0011 1101 | 0100 0010 |

Binary comparison:

```
1's Complement of (A)    10111110  11000010  10111101
                  (B)    01000001  00111101  01000010
                         ----------------------------
                         11111111  11111111  11111111   Partial Sum
                                                    1   Initial Carry
                         ----------------------------
                       1 00000000  00000000  00000000   Final Sum
```

Conditions:

1. Number unequal indicator is turned OFF or remains OFF, since the final sum in the three specified fields is zero.

2. The carry beyond the specified field length and the condition of the number unequal indicator cause the E flag to be set and the repeat indicator to be turned OFF if it is ON.

Example 3. (A) > (B)

Assume that the fields defined in Example 1 contain the following:

| | | | |
|---|---|---|---|
| A | 0200 | 0201 | 0202 |
| (A) | 0100 1110 | 0100 0011 | 0101 0010 |

| | | | |
|---|---|---|---|
| B | 0300 | 0301 | 0302 |
| (B) | 0100 1001 | 0100 0010 | 0100 1101 |

Binary Comparison:

```
1's Complement of (A)  10110001   10111100   10101101
                  (B)  01001001   01000010   01001101
                       ------------------------------
                       11111000   11111110   11100000   Partial Sum
                              1                11 1     Carries
                                                  1     Initial Carry
                       ------------------------------
                       11111010   11111110   11111011   Final Sum
```

Conditions:

1. Number unequal indicator ON (final sum ≠ 0).

2. No carry beyond the specified field. The G flag is set ON and the repeat indicator is turned OFF, if it is ON.

## C = 102(66)(E6)

The REPEAT command sets up conditions that cause the command immediately following it to be repeated n times (repeat number) or skipped (if n = 0). The repeat number is the contents of a 1-character field specified by the effective A operand, and may range in value from 0 through 255, with 0 = 0 in this case. Any command may follow a REPEAT command, but certain commands nullify repeat preparation.

The REPEAT command moves the repeat number into a counter. If the repeat number is not equal to zero, the repeat indicator is turned ON, and the command terminates. If the repeat number is equal to zero, the repeat indicator is not turned on, and the control register is incremented to access the second command following the REPEAT command.

The T and B portions of the command are not used. If a 2-address command is specified, the new T and B values are stored in the appropriate registers.

Near the completion of the setup phase for each command, the repeat indicator is tested. If a PACK, UNPACK, ADD, SUBTRACT, MVAR, or COMPARE command is being set up at the time, and the repeat indicator is ON, the contents of the control register are left unchanged so that the command being repeated will be referenced, and the repeating flow is entered.

In the repeating flow, the repeat counter is decremented by 1 and compared to 0. If the result is not equal, the command being repeated is performed. If the result is equal, the repeat indicator is turned OFF, the control register is incremented to contain the address of the next command, and the command being repeated is performed for the last time. When the repeating flow is entered, the contents of the repeat counter are compared to 0 before decrementation. If the result is equality, a program error (PE) is indicated.

If a BRANCH, WAIT, or INOUT command follows a REPEAT command, and the repeat number is greater than 0, the repeat indicator is turned OFF, nullifying the REPEAT command. If the repeat number is equal to 0, the command following is skipped.

If a REPEAT command is followed by a second REPEAT command, and the repeat number of the first command is not equal to 0, the first command is nullified, and the second is performed. If the repeat number of the first command is equal to 0, the second REPEAT command is skipped.

### Command Execution Time

E = 16 if repeat number $\neq$ 0
E = 20 if repeat number = 0

NOTE

For commands that are repeated:

$$E = n(E) + 5 + 2n$$

C = 103(67)(E7)

The WAIT command causes program operation to stop functionally. Input and output operations that have already been initiated are allowed to continue until completion, however, and, in the user state, I/O termination interrupt is permitted.

The repeat indicator is turned OFF by the command, and the eight least-significant bits of effective A address are displayed on the console, provided that the INFORMATION SELECT switch is in the WAIT position and no ME, PE, or TI exists. All console functions are activated, except LOAD and RESET; the HALT switch must be ON for these operations.

In the wait state, the COMPUTE indicator is tested. If it is OFF, the execution of the WAIT command is continuously repeated, with BCT performed between each repetition. If the indicator is ON, the control register is incremented to access the next command in sequence, and the WAIT command terminates.

In the user state, if the TI flag is ON, control is transferred to the supervisor state, and the I/O termination trap is entered. If the contents of the user sequence control register are not changed by the trap, the WAIT command continues when control returns to the user program. If the contents of the user sequence control register are changed by the trap, operation continues from the new address when control is returned to the user program.

The T and B values of the WAIT command are not used. If a 2-address command is specified, the T and B values are stored in the appropriate registers for subsequent implied T and B operation.

Command Execution Time

E = 5

| Command | | Code |
|---|---|---|
| BRANCH OVERFLOW | (BROV) | 104(68)(E8) |
| BRANCH LESS | (BRL) | 105(69)(E9) |
| BRANCH EQUAL | (BRE) | 106(6A)(EA) |
| BRANCH LESS OR EQUAL | (BRLE) | 107(6B)(EB) |
| BRANCH GREATER | (BRG) | 108(6C)(EC) |
| BRANCH LESS OR GREATER | (BRU) | 109(6D)(ED) |
| BRANCH GREATER OR EQUAL | (BRGE) | 110(6E)(EE) |
| BRANCH UNCONDITIONALLY | (BR) | 111(6F)(EF) |

Each branch command (except BRANCH UNCONDITIONALLY) tests its appropriate flag (G, L, E, or OF). If the flag is OFF, the next command in sequence is performed. The BRANCH UNCONDITIONALLY Command takes the branch in all cases.

## Supervisor State

If the appropriate flag is ON, or a BRANCH UNCONDITIONALLY command is specified, in the supervisor state, the effective A address is stored in the supervisor control register. If b16 of the effective A address is 0, the address of the next command is taken from the supervisor control register (effective A address). If b16 of the effective A address is 1, it is replaced with 0 before storage; the S flag is turned OFF, and the next command address is taken from the user sequence control register.

If the tested flag is OFF, the command following the branch command is performed.

## User State

If the appropriate flag is ON, or a BRANCH UNCONDITIONALLY command is specified, in the user state, the effective A address is stored in the user sequence control register. The next command performed is then the command specified by the effective A address. If b16 of A is 1, it is not turned off before storage, and attempting to execute the command results in a PE.

If the tested flag is OFF, the command following the branch command is performed.

When the BRANCH OVERFLOW command is executed, the OF is turned OFF in both states. The repeat indicator is turned OFF in all cases.

The T and B portion of any branch command are not used. If a 2-address command is specified, the T and B values are stored in the appropriate registers where they are available for subsequent implied T and B operations.

## Command Execution Time

E = 9 for all branch commands

C = 112(70) (FO)

The INOUT command initiates input and output operations only. That is, it selects the peripheral and specifies the function to be performed. Prior to completion of the execution phase, the peripheral sends control characters to the processor indicating the result of the attempted selection.

When the execution phase of the INOUT command has been successfully completed, the I/O control takes charge of data transfer on a memory cycle stealing basis, and the ALU continues processing with the next command in sequence.

For a detailed explanation of NCR Century I/O operations, refer to the I/O Control section of the NCR CENTURY 100 PROCESSOR manual.

Command Execution Time

E (minimum) = 15P-5, where P = No. of PAF characters

E (maximum) = 51P-5, when peripheral times out just before INOUT P-count runs out (see NCR CENTURY 100 PROCESSOR publication)

## TIMING FOR HARDWARE COMMANDS

The introduction to this publication contains formulas for determining total hardware command time in machine cycles. The following examples are offered as an aid to using the formulas.

Example 1. One-address command, no indexing required.

| Q | RA | A2 | A1 |
|---|---|---|---|
| E5 | 00 | 00 | 2A |

CC for 1-address COMPARE BINARY command

According to the basic formula:

M (Total command time) = S(etup time) + E(xecution time)

For a 1-address command with no indexing, the formula

$$S = 10 + 8R_A + 1_A R_A + K(10 + 8R_B + 1_B R_B)$$

becomes:

$$S = 10 + 0 + 0 + 0 = 10$$

E for a COMPARE BINARY command is defined as 11 + 9T
Assume that T = 2, derived from some previous command.

Then:

$$M = 10 + (11 + 18) = 39 \text{ memory cycles}$$

39 memory cycles x 800 nanoseconds/cycle = 31,200 nanoseconds = 31.2 microseconds

Example 2. Two-address command, indexing required for both operands.

| Q | RA | A2 | A1 | T | RB | B2 | B1 |
|---|---|---|---|---|---|---|---|
| 60 | 58 | 08 | 00 | 03 | 30 | 00 | 00 |

CC for 2-address ADD BINARY command

For a 2-address command with Mode 0 indexing specified for both operands,

$$S = 10 + 8R_A + 1_A R_A + K(10 + 8R_B + 1_B R_B)$$

becomes:

$$\begin{aligned} S &= 10 + 8 + 0 + 1(10+8+0) \\ &= 36 \end{aligned}$$

For the given command:

$$E = 9 + 9T = 9 + 9(3)$$

$$M = 36 + [9 + 9(3)] = 72 \text{ cycles}$$

72 cycles x 800 nanoseconds/cycle 57,600 nanoseconds

57.6 microseconds

If incremental indexing had been specified for both operands, then S would have been equal to:

10 + 8 + 1 + 1(10 + 8 + 1)

or 38, and M would have equaled:

38 + [9 + 9(3)]or 74 cycles, for a total command time of 59.2 microseconds.