REFERENCE MANUAL

# CPU INSTRUCTIONS

SYSTEM TEN COMPUTER BY SINGER

Read
Write

Add
Subtract
Multiply
Divide

Move Character
Move Numeric
Exchange

Edit
Form Numeric
Compare
Branch

SINGER
BUSINESS MACHINES

REFERENCE MANUAL

# CPU INSTRUCTIONS

SYSTEM TEN COMPUTER BY SINGER

Publication No. 42-2500-01
April 1973

SINGER
BUSINESS MACHINES
2350 WASHINGTON AVE.
SAN LEANDRO, CALIF. 94577

PRINTED IN U.S.A.

# CONTENTS

(This page intentionally left blank)

## INSTRUCTION FORMAT

Each System Ten* computer instruction is 10 characters long. Each instruction must be positioned so that the address of the leftmost character is a multiple of 10 (e.g., 0, 10, 20, 30....etc.). The first few characters of an instruction as they appear in memory have the following format:

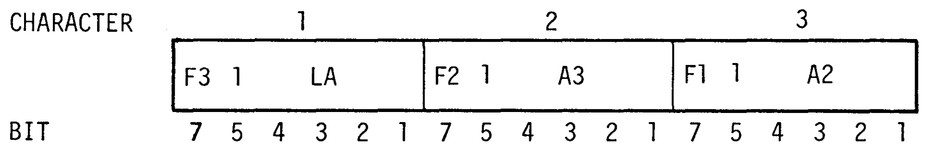


Figure 1 Instruction Format - Sequential

*A trademark of the Singer Company.

A more useful representation of instruction format is achieved by giving a vertical orientation to the bits of a character as is done below.

| CHARACTER ▷ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | BIT ▽ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | F3 | F2 | F1 | F0 | AC | IA1 | IA0 | IB1 | IB0 | BC | 7 |
| | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 5 |
| | | | | | | | | | | | 4 |
| | LA | A3 | A2 | A1 | A0 | LB | B3 | B2 | B1 | B0 | 3 |
| | | | | | | | | | | | 2 |
| | | | | | | | | | | | 1 |

Figure 2 Instruction Format - Parallel Blocking

In this representation functionally related bits such as F3 - F0 also have a close spatial relationship.

## Operation Code

The operation code of an instruction is specified by the four bit binary number F = F3F2F1F0, e.g., an ADD instruction is indicated when F = 0100 and a COMPARE when F = 1110.

## Address Fields

Each instruction contains two address fields A and B. These are generally used to specify the addresses of the two operands which participate in the operation specified by F. Each address is a four digit decimal number between 0000 and 9999 inclusive.

The A-address is given by A3A2A1A0, the numeric portion (bits 1 thru 4) of characters 1 thru 4.

The B-address is given by B3B2B1B0, the numeric portion of characters 6 thru 9.

## Addressing Mode

An instruction address may refer to a location in COMMON or in partition.

| | |
|---|---|
| AC = 1 | Means the A address refers to a location in COMMON. |
| AC = 0 | Means the A address refers to a location in partition. |
| BC = 1 | Means the B address refers to a location in COMMON. |
| BC = 0 | Means the B address refers to a location in partition. |

## Indexing

In most instructions both the A and B address may be indexed. Index register selection for the A address is determined by IA, and for the B address by IB, according to the table below:

| IA1 IB1 | IA0 IB0 | |
|---|---|---|
| 0 | 0 | NO INDEXING |
| 0 | 1 | INDEX REGISTER ONE |
| 1 | 0 | INDEX REGISTER TWO |
| 1 | 1 | INDEX REGISTER THREE |

Table 1 Index Addresses

## Operand Lengths

Operand lengths are explicitly defined using LA and LB, the numeric portion of characters 0 and 5 respectively. Certain instructions use LA and LB differently as will be discussed later.

LA--length, in number of characters of the Operand-A.

LB--length, in number of characters of the Operand-B.

## ADD INSTRUCTION

The Add instruction adds the numeric portions of two operands algebraically. The sum replaces the second operand and leaves the first operand unchanged if the fields do not overlap.

## INSTRUCTION FIELDS

### Machine Operation Code

F---Binary 0100 (4).

### Address Specification

A---Address of the leftmost position of Operand-A.

B---Address of the leftmost position of Operand-B.

### Indexing Specification

IA--Index register for determining effective address of Operand-A.

IB--Index register for determining effective address of Operand-B.

### Common Partition Specification

AC--If AC is 0, A is address in controlling partition.
If AC is 1, A is address in Common.

BC--If BC is 0, B is address in controlling partition.
If BC is 1, B is address in Common.

### Length Specification

LA--Length of Operand-A.

LB--Length of Operand-B.

## OPERAND FIELDS

### Operand-A Address

If IA is 0, then A is the effective address.
If IA is 1, 2, or 3, the corresponding index register is added to A to determine the effective address of Operand-A.

If AC is 1, the effective address lies in Common.

### Operand-B Address

If IB is 0, then B is the effective address.
If IB is 1, 2, or 3, the corresponding index register is added to B to determine the effective address of Operand-B.

If BC is 1, the effective address lies in Common.

### Operand Lengths

If LA is 0, the length of Operand-A is 10 characters.
If LA is 1 thru 9, the length of Operand-A is 1 thru 9 characters.

If LB is 0, the length of Operand-B is 10 characters.
If LB is 1 thru 9, the length of Operand-B is 1 thru 9 characters.

## OPERATION

### General Description

The add operation proceeds from right to left starting with the rightmost characters of Operand-A and Operand-B. Character by character, the algebraic sum is developed in Operand-B.

If Operand-A is shorter than Operand-B, the operation proceeds normally until Operand-A is exhausted. After that, the process continues in similar fashion except that a zero character is automatically substituted every time the adding logic calls for a character from Operand-A. In effect, Operand-A is given enough preceding zeros to make it the same length as Operand-B.

If Operand-A is longer than Operand-B, addition stops after the leftmost position in Operand-B has been added. The remaining positions in Operand-A are ignored, and do not affect the sum or the Condition Code.

The algebraic sign of the sum is placed in bit-7 of the rightmost position of Operand-B, and bit-5 is turned ON. Except for the rightmost character, the other zone bits of Operand-B are unchanged. Operand-A is unchanged by the add operation.

If the sum exceeds the capacity of Operand-B, a carry-to-the-left from the leftmost position does not occur. Condition Code 4 is set to indicate the overflow.

### Condition Codes

After completion of the Add instruction.

1 = Negative, non-zero sum.
2 = Zero sum.
3 = Positive, non-zero sum.
4 = Overflow.

### Execution Time (T) in Microseconds

T = 42.2 + 3.3 (LA) + 10.0 (LB) + TIX + TOD, if LA is equal to or less than LB.

T = 42.2 + 11 (LA) + 12.2 (LB) + TIX + TOD, if LA is greater than LB.

Key: TIX = 0.0, if IA and IB are both zero.
TIX = 58.9, if IA and IB are both non-zero.
TIX = 31.1, if IA <u>or</u> IB is non-zero.

TOD = 0.0, if an overdraft <u>does</u> <u>not</u> occur.
TOD = 10.0 (LB), if an overdraft occurs.

An overdraft will always occur when the absolute value of Operand-A exceeds the absolute value of Operand-B <u>and</u> they have <u>un</u>like signs.

## PROGRAMMING HINTS

### Overlapped Operands

In case of overlapped operands, the result is unspecified.

(This page intentionally left blank)

## BRANCH INSTRUCTION

The Branch instruction permits departure from the sequential path by which instructions are normally executed. Branching can be unconditional, it can depend upon the current status of the Condition Code, or it can depend upon signals from Input/Output devices requesting service from the CPU. A variant of the Branch instruction passes control to a subroutine after first setting the return address at which the main program will be resumed. Execution of the Branch instruction does not alter the Condition Code.

## INSTRUCTION FIELDS

### Machine Operation Code

F---Binary 1011 (11).

### Address Specification

A---Address-A

B---Address-B

### Indexing Specification

IA--Ignored. Branch instructions are not indexed.

IB--Ignored. Branch instructions are not indexed.

### Common Partition Specification

AC--If AC is 0, A is an address in controlling partition.
If AC is 1, A is an address in Common.

BC--If BC is 0, B is an address in controlling partition.
If BC is 1, B is an address in Common.

### Variant Specification

LA--A digit 0-9.

LB--A digit 0-6, 8, 9.

## OPERATION

### Order of Presentation

The Branch instruction consists of several variants. The LA and LB instruction fields determine which variant is executed. "Link" (variant 6) and "Branch on Service Request" (variant 7) require that the entire instruction be decoded. These variants are discussed later under separate headings. The other variants are decoded and executed a half instruction at a time and are most conveniently discussed as a group in the next paragraph.

### Variants 0, 1, 2, 3, 4, 5, 8, 9

The first five characters of the instruction are fetched. LA is examined. If a branch is required, control passes to Address-A, and the right half of the instruction is ignored. If a branch is not required in the left half of the instruction, the right half is fetched. LB is examined. If a branch is required, control passes to Address-B. If a branch is not required, execution continues with the next sequential instruction.

The following table shows the values which LA and LB may assume. Beside each variant number is the meaning applied by the ACU. Variant 6 and variant 7 are purposely omitted. They are discussed under "Link" and "Branch on Service Request".

Variant 0---Do not branch ("no operation").

Variant 1---Branch if Condition Code is 1.

Variant 2---Branch if Condition Code is 2.

Variant 3---Branch if Condition Code is 3.

Variant 4---Branch if Condition Code is 4.

Variant 5---Branch, unconditionally.

Variant 8---Branch and switch partitions, unconditionally.

Variant 9---Do not branch ("no operation").

### Partition Switching

If a Branch instruction does not require a branch, execution simply continues with the next sequential instruction.

If the host partition has been in continuous control for more than 37.5 milliseconds when a branch is required, the branch is taken but the execution of the instruction at the branch address is postponed and control passes to the next partition. When control returns, execution resumes at the branch address. If the branch is caused by variant 8 ("Branch and switch, unconditionally"), the branch is taken but the execution of the instruction at the branch address is postponed and control passes to the next partition even though 37.5 milliseconds have not elapsed.

## LINK - BRANCH VARIANT 6

LA--Must be 6.

LB--May be 0 thru 5, 8, or 9.

If LB is 0 or 9, no link occurs; control simply passes to the next instruction.

If LB is 1-4, the corresponding Condition Code is tested. If the specified Condition Code is ON, the link operation is performed. Otherwise, control simply passes to the next instruction.

If LB is 5 or 8, the link operation is performed, unconditionally.

### Return Address/Start Address

The address of the next instruction (return address) is inserted into the numerical portion of the four position field starting at Address-A. The zone portions of the three left character positions are unchanged. Bit-5 of the rightmost position is set to 1. Bit-7 is set to 1 if the return address is in common; it is set to 0 if the return address is in partition. Control then passes to Address-B (start address).

## BRANCH ON SERVICE REQUEST - BRANCH VARIANT 7

LA--Must be 7.

LB--Must be 0 or 9.

| INSTRUCTION | CC = 1 | CC = 2 | CC = 3 | CC = 4 |
|---|---|---|---|---|
| $\boxed{A}$DD | MINUS | ZERO | PLUS | OVERFLOW |
| $\boxed{B}$RANCH $\boxed{C}$ONDITIONAL | ——— | ——— | ——— | ——— |
| $\boxed{C}$OMPARE | A IS LESS | EQUAL | A IS GREATER | A NOT LESS |
| $\boxed{D}$IVIDE | MINUS | ZERO | PLUS | OVERFLOW |
| $\boxed{E}$DIT | MINUS | ZERO | PLUS | ——— |
| E$\boxed{X}$CHANGE | ——— | 2 ALWAYS SET | ——— | ——— |
| $\boxed{F}$ORM $\boxed{N}$UMERIC FIELD | MINUS | ZERO | PLUS | OVERFLOW |
| $\boxed{M}$OVE $\boxed{C}$HARACTER | ——— | 2 ALWAYS SET | ——— | ——— |
| $\boxed{M}$OVE $\boxed{N}$UMERIC | ——— | 2 ALWAYS SET | ——— | ——— |
| $\boxed{M}$ULTIPLY | MINUS | ZERO | PLUS | ——— |
| $\boxed{R}$EAD | ERROR | NORMAL | FLAG | FAULT |
| $\boxed{S}$UBTRACT | MINUS | ZERO | PLUS | OVERFLOW |
| $\boxed{W}$RITE | ERROR | NORMAL | FLAG | FAULT |

Table 2 Condition Code Settings

### Operation - Storing Device Number

Each IOC continually polls the input/output devices attached to it to see if a device has signalled a request for service. If the IOC encounters such a signal, further polling for service requests is temporarily discontinued, and the device number is held in a counter until the CPU executes "Branch on Service Request". "Branch on Service Request" causes the counter to be stored in the numeric portion of the character position pointed to by Address-A. Control then passes to Address-B. Polling resumes with the next higher device number (or 0, if the requesting device was 9).

If the IOC is holding no such request for service, "Branch on Service Request" has no effect. Execution continues with the next sequential instruction.

### Condition Codes

Condition Codes are unchanged by the Branch instruction.

### Execution Time (T) in Microseconds

T = 37.8 for no branch.
T = 27.8 for branch to Address-A.
T = 44.4 for branch to Address-B (except variants 6,7).
T = 75.5 for "Link" (variant 6).
T = 51.1 for "Branch on Service Request" (variant 7).

## PROGRAMMING HINTS

Since each instruction (with the exception of Branch) sets the condition code, it is necessary to test the condition code immediately after the performance of an operation.

(This page intentionally left blank)

## COMPARE INSTRUCTION

The Compare instruction compares two fields and sets the Condition Code to indicate the relation between them.

## INSTRUCTION FIELDS

### Machine Operation Code

F---Binary 1110 (14).

### Address Specification

A---Address of the leftmost position of Operand-A.

B---Address of the leftmost position of Operand-B.

### Indexing Specification

IA--Index register for determining effective address of Operand-A.

IB--Index register for determining effective address of Operand-B.

### Common Partition Specification

AC--If AC is 0, A is address in controlling partition.
If AC is 1, A is address in Common.

BC--If BC is 0, B is address in controlling partition.
If BC is 1, B is address in Common.

### Length Specification

LA--Tens position of length of both Operand-A and Operand-B.

LB--Units position of length of both Operand-A and Operand-B.

## OPERAND FIELDS

### Operand-A Address

If IA is 0, then A is the effective address.
If IA is 1, 2, or 3, the corresponding index register is added to A to determine the effective address of Operand-A.

If AC is 1, the effective address lies in Common.

### Operand-B Address

If IB is 0, then B is the effective address.
If IB is 1, 2, or 3, then corresponding index register is added to B to determine the effective address of Operand-B.

If BC is 1, the effective address lies in Common.

### Operand Lengths

Operand-A and Operand-B are equal in length.
10LA + LB = Lengths of operands for the Compare instruction.
If 10LA + LB = 00, 100 is the length of the operands.

## OPERATION

### General Description

The compare operation proceeds from left to right starting with the leftmost character of Operand-A and Operand-B. Character by character, the values of Operand-A and Operand-B are compared until a difference is found <u>or</u> the rightmost position has been compared.

When the characters differ, Condition Code 1, <u>or</u> 3 and 4 is set ON (indicating that Operand-A is smaller <u>or</u> larger than Operand-B), and the operation is complete.

If the characters are identical, and there are more positions to be compared, the comparison is repeated for the next position on the right.

When the characters are identical and there are no more positions to be compared, Condition Codes 2 and 4 are set ON.

Operand-A and Operand-B are unchanged by the compare operation.

When Condition Code 3 or 2 is set ON, Condition 4 is also set ON.

### Condition Codes:

1, if Operand-A is less than Operand-B.
2 and 4, if Operand-A and Operand-B are identical.
3 and 4, if Operand-A is greater than Operand-B.

### Execution Time (T) is Microseconds

T = 40.0 + 7.78 (10LA + LB) + TIX, if the operands are identical.
T = 48.9 + 7.78 (Y) + TIX, if the operands differ.

```
    Key:  Y = the number of equal comparisons.
        TIX =  0.0, if IA and IB are both zero.
        TIX = 58.9, if IA and IB are both non-zero.
        TIX = 31.1, if IA or  IB is non-zero.
```

## PROGRAMMING HINTS

### Character Values

The reader is referred to the Table 3 entitled "Characters Arranged in Sequence of Value." In the first column under "Character Code" are the internal codes of each character used in the Model 20 Processor. In the second column under "Character" are the corresponding characters. The table can be used to resolve uncertainties as to which of two characters the Compare instruction considers to be the larger. A character is considered greater than the other characters which precede it in the table. It is less than those which follow it.

### Sorting

A principal use of the Compare instruction is in sorting data. The programmer is reminded that the units position of a negative numeric field is coded with zone bit-7 ON. (If the digit were positive, bit-7 would be OFF.) Thus, in a compare operation, a negative digit is of greater value than any positive digit.

| Character Code | | | | | | Character | |
|---|---|---|---|---|---|---|---|
| $b_7$ | $b_5$ | $b_4$ | $b_3$ | $b_2$ | $b_1$ | | |
| 0 | 0 | 0 | 0 | 0 | 0 | SP | Space |
| 0 | 0 | 0 | 0 | 0 | 1 | ! | Exclamation Point |
| 0 | 0 | 0 | 0 | 1 | 0 | " | Quotation Mark |
| 0 | 0 | 0 | 0 | 1 | 1 | # | Number Sign |
| 0 | 0 | 0 | 1 | 0 | 0 | $ | Dollar Sign |
| 0 | 0 | 0 | 1 | 0 | 1 | % | Percent |
| 0 | 0 | 0 | 1 | 1 | 0 | & | Ampersand |
| 0 | 0 | 0 | 1 | 1 | 1 | ' | Prime, Apostrophe |
| 0 | 0 | 1 | 0 | 0 | 0 | ( | Left Parenthesis |
| 0 | 0 | 1 | 0 | 0 | 1 | ) | Right Parenthesis |
| 0 | 0 | 1 | 0 | 1 | 0 | * | Asterisk |
| 0 | 0 | 1 | 0 | 1 | 1 | + | Plus Sign |
| 0 | 0 | 1 | 1 | 0 | 0 | , | Comma |
| 0 | 0 | 1 | 1 | 0 | 1 | - | Minus Sign, Hyphen |
| 0 | 0 | 1 | 1 | 1 | 0 | . | Period, Decimal Point |
| 0 | 0 | 1 | 1 | 1 | 1 | / | Slash |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | Zero |
| 0 | 1 | 0 | 0 | 0 | 1 | 1 | One |
| 0 | 1 | 0 | 0 | 1 | 0 | 2 | Two |
| 0 | 1 | 0 | 0 | 1 | 1 | 3 | Three |
| 0 | 1 | 0 | 1 | 0 | 0 | 4 | Four |
| 0 | 1 | 0 | 1 | 0 | 1 | 5 | Five |
| 0 | 1 | 0 | 1 | 1 | 0 | 6 | Six |
| 0 | 1 | 0 | 1 | 1 | 1 | 7 | Seven |
| 0 | 1 | 1 | 0 | 0 | 0 | 8 | Eight |
| 0 | 1 | 1 | 0 | 0 | 1 | 9 | Nine |
| 0 | 1 | 1 | 0 | 1 | 0 | : | Colon |
| 0 | 1 | 1 | 0 | 1 | 1 | ; | Semicolon |
| 0 | 1 | 1 | 1 | 0 | 0 | < | Less-than Sign |
| 0 | 1 | 1 | 1 | 0 | 1 | = | Equal Sign |
| 0 | 1 | 1 | 1 | 1 | 0 | > | Greater-than Sign |
| 0 | 1 | 1 | 1 | 1 | 1 | ? | Question Mark |
| 1 | 0 | 0 | 0 | 0 | 0 | @ | At Sign |
| 1 | 0 | 0 | 0 | 0 | 1 | A | |
| 1 | 0 | 0 | 0 | 1 | 0 | B | |
| 1 | 0 | 0 | 0 | 1 | 1 | C | |
| 1 | 0 | 0 | 1 | 0 | 0 | D | |
| 1 | 0 | 0 | 1 | 0 | 1 | E | |
| 1 | 0 | 0 | 1 | 1 | 0 | F | |
| 1 | 0 | 0 | 1 | 1 | 1 | G | |
| 1 | 0 | 1 | 0 | 0 | 0 | H | |
| 1 | 0 | 1 | 0 | 0 | 1 | I | |
| 1 | 0 | 1 | 0 | 1 | 0 | J | |
| 1 | 0 | 1 | 0 | 1 | 1 | K | |
| 1 | 0 | 1 | 1 | 0 | 0 | L | |
| 1 | 0 | 1 | 1 | 0 | 1 | M | |
| 1 | 0 | 1 | 1 | 1 | 0 | N | |
| 1 | 0 | 1 | 1 | 1 | 1 | O | |
| 1 | 1 | 0 | 0 | 0 | 0 | P | |
| 1 | 1 | 0 | 0 | 0 | 1 | Q | |
| 1 | 1 | 0 | 0 | 1 | 0 | R | |
| 1 | 1 | 0 | 0 | 1 | 1 | S | |
| 1 | 1 | 0 | 1 | 0 | 0 | T | |
| 1 | 1 | 0 | 1 | 0 | 1 | U | |
| 1 | 1 | 0 | 1 | 1 | 0 | V | |
| 1 | 1 | 0 | 1 | 1 | 1 | W | |
| 1 | 1 | 1 | 0 | 0 | 0 | X | |
| 1 | 1 | 1 | 0 | 0 | 1 | Y | |
| 1 | 1 | 1 | 0 | 1 | 0 | Z | |
| 1 | 1 | 1 | 0 | 1 | 1 | [ | Opening Bracket |
| 1 | 1 | 1 | 1 | 0 | 0 | \ | Reverse Slant |
| 1 | 1 | 1 | 1 | 0 | 1 | ] | Closing Bracket |
| 1 | 1 | 1 | 1 | 1 | 0 | ^ | Circumflex |
| 1 | 1 | 1 | 1 | 1 | 1 | — | Underline |

CHARACTERS ARRANGED
IN SEQUENCE OF VALUE

Table 3 Characters Arranged in Sequence of Value

## DIVIDE INSTRUCTION

The Divide instruction computes the algebraic quotient (and remainder) of two operands.

## INSTRUCTION FIELDS

### Machine Operation Code

F---Binary 0101 (5).

### Address Specification

A---Address of the leftmost position of Operand-A.

B---Address of the leftmost position of Operand-B (dividend) Address of the quotient.

### Indexing Specification

IA--Index register for determining effective address of Operand-A.

IB--Index register for determining effective address of Operand-B.

### Common Partition Specification

AC--If AC is 0, A is an address in controlling partition.
If AC is 1, A is an address in Common.

BC--If BC is 0, B is an address in controlling partition.
If BC is 1, B is an address in Common.

### Length Specification

LA--Length of Operand-A (divisor).

LB--Length of the quotient.

LA + LB--Length of Operand-B (dividend).

## OPERAND FIELDS

### Operand-A Address

If IA is 0, then A is the effective address.
If IA is 1, 2, or 3, the corresponding index register is added to A to determine the effective address of Operand-A.

If AC is 1, the effective address lies in Common.

### Operand-B Address

If IB is 0, then B is the effective address.
If IB is 1, 2, or 3, the corresponding index register is added to B to determine the effective address of Operand-B.

If BC is 1, the effective address lies in Common.

### Operand Lengths

If LA is 0, the length of Operand-A is 10 characters.
If LA is 1 thru 9, the length of Operand-A is 1 thru 9 characters.

If LB is 0, the length of Quotient is 10 characters.
If LB is 1 thru 9, the length of Quotient is 1 thru 9 characters.

LA + LB is the length of Operand-B (dividend).

## OPERATION

### General Description

Operand-A is the divisor.

The dividend begins at the B address and contains LB + LA positions.

At the end of the operation, the quotient occupies the leftmost LB positions of the dividend field, and the remainder occupies the rightmost LA positions of the dividend field.

If the divisor and the dividend differ in sign, bit-7 of the quotient is turned ON to indicate a negative quotient. If the signs are alike, bit-7 is turned OFF to indicate a positive quotient. Bit-5 is turned ON for all positions of the quotient; bit-7 is turned OFF for all positions except the rightmost.

Bit-7 of the rightmost position of the remainder is unchanged. It continues to show the sign of the dividend. Bit-5 is set to 1. The zone bits of the other positions in the remainder are unchanged.

## Process

An internal counter is set to zero. It will count the number of times the divisor is subtracted from a subfield-of-the-dividend. The subfield length is one greater than the length of the divisor. The first subfield chosen is at the extreme left of the dividend.

The divisor is repeatedly subtracted from the subfield until the value of the subfield is less than that of the divisor. Each subtraction increments the counter. If the count exceeds 9, Condition Code 4 is set (indicating overflow), and the operation is abandoned. If the count does not exceed 9, and the subfield value is less than the divisor, the count is stored in the leftmost position of the subfield where it is also the leftmost position of the quotient. The counter is cleared, and the process shifts to the next subfield (one character position to the right in the dividend) to develop the second position of the quotient. After this, another shift to develop the third position, etc. The operation ends after the rightmost subfield in the dividend is processed in this fashion.

## Condition Codes

After completion of the Divide instruction:

| | |
|---|---|
| 1 = | Negative, non-zero quotient. |
| 2 = | Zero quotient. |
| 3 = | Positive, non-zero quotient. |
| 4 = | Overflow. |

## Execution Time (T) in Microseconds

T = 46.67 + 1.11 (LA) + 26.67 (LB) + 22.22 (LA) (LB) + (10.0 + 11.1 (LA)) (S) + TIX.

Key: TIX = 0.0, if IA and IB are both zero.
TIX = 58.9, if IA and IB are both non-zero.
TIX = 31.1, if IA <u>or</u> IB is non-zero.

S = Sum of digits in quotient.

## PROGRAMMING HINTS

### Overlapped Operands

In case of overlapped operands, the result is unspecified.

### Division by Zero

An attempt to divide by zero causes Condition Code 4 to be set (indicating overflow). The value of the dividend is unchanged.

### Preventing Overflow

Overflow will only occur if the absolute value in the leftmost LA positions of the dividend equals or exceeds the absolute value of the divisor. In cases where it is necessary to accommodate the widest possible range of data, including division by 1, the leftmost LA positions of the dividend should contain zero.

## EDIT INSTRUCTION

The Edit instruction moves a 1-100 digit numerical field into a "control" field so that the information is in a form suitable for printing. The control field governs the suppression of preceding zeros (including the insertion of check protection characters ahead of significant digits), the insertion of punctuation marks, and the indication of sign.

## INSTRUCTION FIELDS

### Machine Operation Code

F---Binary 1100 (12).

### Address Specification

A---Address of the leftmost position of Operand-A.

B---Address of the leftmost position of Operand-B.

### Indexing Specification

IA--Index register for determining effective address of Operand-A.

IB--Index register for determining effective address of Operand-B.

### Common Partition Specification

AC--If AC is 0, A is an address in controlling partition.
If AC is 1, A is an address in Common.

BC--If BC is 0, B is an address in controlling partition.
If BC is 1, B is an address in Common.

### Length Specification

LA--Tens position of length of Operand-A.

LB--Units position of length of Operand-A.

## OPERAND FIELDS

### Operand-A Address

If IA is 0, then A is the effective address.
If IA is 1, 2, or 3, the corresponding index register is added to A to determine the effective address of Operand-A.

If AC is 1, the effective address lies in Common.

### Operand-B Address

If IB is 0, then B is the effective address.
If IB is 1, 2, or 3, the corresponding index register is added to B to determine the effective address of Operand-B.

If BC is 1, the effective address lies in Common.

### Operand Lengths

The length of Operand-A is (10)LA+LB.
If (10)LA+LB = 00, the length = 100.

The length of Operand-B is the sum of the following:

Operand-A length + 1.

The number of punctuation characters in Operand-B.

The number of @ characters in Operand-B.

## OPERATION

### Operand-B, the Control Field

A filler character is defined as any character other than the @ sign or a punctuation mark (comma, decimal point, hyphen, slash).

Minimally, a control field consists of as many filler characters as there are digits in Operand-A plus one trailing character to show sign. In addition, the filler characters may be freely interspersed with punctuation characters (comma, period, hyphen, slash) and @ signs.

Since the Edit instruction destroys the control field, the programmer normally moves the control field to the Operand-B address before each use of the Edit instruction.

The filler characters designate the mask positions into which Operand-A digits can be moved. Significant digits from Operand-A simply replace the corresponding filler positions in the control field. Filler characters corresponding to non-significant zeros in Operand-A are not replaced, they are undisturbed. This permits the suppression of preceding zeros (i.e., the filler positions are preset to contain blank characters) or the use of check protection characters ahead of significant digits (i.e., the filler positions are preset to contain a protect character such as asterisk).

The punctuation characters are used to punctuate the significant information received from Operand-A. At the completion of the Edit instruction, any punctuation characters which find themselves embedded in the significant portion of the control field remain undisturbed by the operation and thus show the desired punctuation. Any punctuation character to the left of the significant portion of the control field will have been replaced by the neighboring character on the left and thus wiped out. A control field should not begin with a punctuation character.

The @ sign is used to insert blank characters between filler positions. Execution of the Edit instruction replaces each @ sign in the mask with a blank character.

The rightmost position of the control field is used to show the sign of Operand-A. Ordinarily, the programmer presets the position to contain a hyphen or some other character to indicate minus. If Operand-A is negative, the minus character remains. If Operand-A is zero or positive, the minus character is overwritten with a blank character.

## Execution of Edit Instruction

The Edit instruction begins by extracting the leftmost digit of Operand-A and by finding the leftmost filler character in the control field. During the hunt for the filler character, any intervening @ sign in the control field is replaced by a blank character, and any intervening punctuation mark is replaced by the neighboring character on the left.

If the Operand-A digit is significant, the numeric portion is put into the filler position of the control field, and the zone bits of that position are set to 0/1 to insure that the position will print as a numerical value.

If the digit is non-significant zero, but the filler character is 0, the digit is stored in the filler position as a significant zero (as are any to the right of it in Operand-A).

If the digit is non-significant, the filler character is left undisturbed.

The process is repeated using the next digit to the right in Operand-A and the next filler character in the control field. Once a significant digit has been moved from Operand-A into the control field, any punctuation mark to the right of it is allowed to stand and is not replaced by its left-hand neighbor.

The process continues until the rightmost digit in Operand-A and the rightmost filler character of the control field have been dealt with. The Condition Code is set. If Operand-A contains a positive value or zero, a blank character is set in the sign position of the control field (the position just to the right of the rightmost filler character).

## Condition Codes

After completion of the Edit instruction.

| | |
|---|---|
| 1 | Negative, non-zero Operand-A. |
| 2 | Zero Operand-A. |
| 3 | Positive, non-zero Operand-A. |

An overflow condition is not possible.

## Execution Time (T) in Microseconds

```
T  41.1 + 10.0 (LA + LB) + 6.67 (X1) + 3.33 (X2) +
    2.22 (X3) + 2.22 (X4) + TIX.

    Key X1 = Number of '@' signs  in control field plus
             number of periods (.), commas (,),   slash
             (/), and minus (-)  signs before  signifi-
             cance in Operand-B control field.

        X2 = Number of periods (.), commas (,), slash (/),
             and minus (-) signs after significance  in
             Operand-B control field.

        X3 = Number of significant  digits in Operand-A.

        X4 = 0 for a negative operand.
             1 for a positive operand.

        TIX =  0.0, if IA and IB are both zero.
        TIX = 58.9, if IA and IB are both non-zero.
        TIX = 31.1, if IA or  IB is non-zero.
```

## EXAMPLES

### Printing Social Security Numbers

```
Operand-A      098144159
Operand-B      000-00-0000-         before editing
Operand-B      098-14-4159          after editing
```

### Check Protection

```
Operand-A      0000001234
Operand-B      **,***,***.00-      before editing
Operand-B      *******12.34        after editing
```

### Use of Commas

```
Operand-A      1234567890
Operand-B      bb,bbb,bbb.00-       before editing
Operand-B      12,345,678.90        after editing

Note---b is here used to represent a blank character.
```

### Suppressing Preceding Zeros

```
Operand-A      0000012345
Operand-B      bb,bbb,bbb.00-       before editing
Operand-B      bbbbbbb123.45        after editing

Note---b is here used to represent a blank character.
```

## EXCHANGE INSTRUCTION

The Exchange instruction interchanges the characters in two fields of equal length in main memory. Each field can comprise 1 - 100 characters.

## INSTRUCTION FIELDS

### Machine Operation Code

F---Binary 1111 (15).

### Address Specification

A---Address of the leftmost position of Operand-A.

B---Address of the leftmost position of Operand-B.

### Indexing Specification

IA--Index register for determining effective address of Operand-A.

IB--Index register for determining effective address of Operand-B.

### Common Partition Specification

AC--If AC is 0, A is an address in controlling partition. If AC is 1, A is an address in Common.

BC--If BC is 0, B is an address in controlling partition. If BC is 1, B is an address in Common.

### Length Specification

LA--Tens position of length of both Operand-A and Operand-B.

LB--Units position of length of both Operand-A and Operand-B.

## OPERAND FIELDS

### Operand-A Address

If IA is 0, then A is the effective address.
If IA is 1, 2, or 3, the corresponding index register is added to A to determine the effective address of Operand-A.

If AC is 1, the effective address lies in Common.

### Operand-B Address

If IB is 0, then B is the effective address.
If IB is 1, 2, or 3, the corresponding index register is added to B to determine the effective address of Operand-B.

If BC is 1, the effective address lies in Common.

### Operand Lengths

Operand-A and Operand-B are equal in length.
10LA + LB = Lengths of operands for Move Character instruction.
If 10LA + LB = 00, 100 is the length of the operands.

## OPERATION

### General Description

The leftmost character of Operand-B is extracted and held temporarily in a register. The character in the leftmost position of Operand-A is moved to the leftmost position in Operand-B, and the character in the register is then stored in the leftmost position of Operand-A. This operation is repeated from left to right until the entire fields have been interchanged.

### Condition Code

2, after completion of the Exchange instruction.

### Execution Time (T) in Microseconds

T = 38.9 + 13.3 (10LA + LB) + TIX.

| | | |
|---|---|---|
| Key: | TIX = 0.0 | if IA and IB are both zero |
| | TIX = 58.9 | if IA and IB are both non-zero |
| | TIX = 31.1 | if IA or IB is non-zero. |

## PROGRAMMING HINTS

If Operand-A and Operand-B do not overlap, a simple exchange occurs.

If Operand-A and Operand-B overlap each other, the programmer can predict the result for any particular case by mentally stepping through the operation as described in "General Description" above.

NOTE----Using an overlapped exchange instruction can be useful for rotating characters of a field. If Operand-A and Operand-B overlap for all but one character, then each time the exchange instruction is executed the leftmost character moves to the rightmost position, and all other characters move one position to the left.

(This page intentionally left blank)

## FORM NUMERIC

The Form Numeric instruction moves numeric information from a 1-10 position mixed field to a second 1-10 position field. After the operation, the second field is of the numerical form normally used for arithmetic operations.

## INSTRUCTION FIELDS

### Machine Operation Code

F---Binary 1101 (13).

### Address Specification

A---Address of the leftmost position of Operand-A.

B---Address of the leftmost position of Operand-B.

### Indexing Specification

IA--Index register for determining effective address of Operand-A.

IB--Index register for determining effective address of Operand-B.

### Common Partition Specification

AC--If AC is 0, A is an address in controlling partition.
If AC is 1, A is an address in Common.

BC--If BC is 0, B is an address in controlling partition.
If BD is 1, B is an address in Common.

### Length Specification

LA--Length of Operand-A.

LB--Length of Operand-B.

## OPERAND FIELDS

### Operand-A Address

If IA is 0, then A is the effective address.
If IA is 1, 2, or 3, the corresponding index register is added to A to determine the effective address of Operand-A.

If AC is 1, the effective address lies in Common.

### Operand-B Address

If IB is 0, then B is the effective address.
If IB is 1, 2, or 3, the corresponding index register is added to B to determine the effective address of Operand-B.

If BC is 1, the effective address lies in Common.

### Operand Lengths

If LA is 0, the length of Operand-A is 10 characters.
If LA is 1 thru 9, the length of Operand-A is 1 thru 9 characters.

If LB is 0, the length of Operand-B is 10 characters.
If LB is 1 thru 9, the length of Operand-B is 1 thru 9 characters.

## OPERATION

### Execution of Form Numeric Instruction

Execution begins with a right-to-left search for the rightmost digit in Operand-A and a determination of its sign:

----If the rightmost non-blank character is a digit, it is moved unchanged into the rightmost position of Operand-B. The sign of Operand-B is positive.

----If the rightmost non-blank character is one of the characters P thru Y, it is considered to be a digit with a minus sign. It is moved unchanged into the rightmost position of Operand-B. The sign of Operand-B is negative.

----If the rightmost non-blank character is a hyphen (minus sign), the rightmost digit is converted to the corresponding character P thru Y (i.e., bit-7 is set ON) and is stored in the rightmost position of Operand-B. The sign of Operand-B is negative.

----If the rightmost non-blank character is none of the above, it is skipped over and the rightmost digit is moved unchanged into the rightmost position of Operand-B. The sign of Operand-B is positive.

Once the rightmost digit is selected from Operand-A and is moved into Operand-B, the process continues from right to left. The next digit to the left is found in Operand-A and is moved unchanged into the next left position of Operand-B. Intervening characters which are not digits are simply passed over and are not moved.

If a digit is moved into the leftmost position of Operand-B and there are yet unmoved digits in Operand-A, the operation is abandoned and Condition Code 4 is set to show the overflow condition.

When the leftmost digit of Operand-A is moved into an Operand-B position, any unfilled positions in Operand-B are set to zero and the operation is finished.

If Operand-A consists entirely of blank characters, no digits can be moved. In this case, Operand-B is set to zero in all positions.

**Condition Codes**

After completion of the Form Numeric instruction.

1 = Negative, non-zero Operand-B.
2 = Zero Operand-B.
3 = Positive, non-zero Operand-B.
4 = Overflow.

**Execution Time (T) in Microseconds**

T = 43.3 + 3.33 (LA) + 7.78 (LB) + 2.22 (Z) + TIX,
if LA - Z is equal to or less than LB.

T = 45.55 + 1.11 (LA) + 10.0 (LB) + 4.44 (Z') + TIX,
if LA - Z is greater than LB, causing an improper overflow.

Key Z = Number of non-numeric characters in Operand-A.

Z' = Number of non-numeric characters encountered i Operand-A before LB is filled.

TIX = 0.0, if IA and IB are both zero.
TIX = 58.9, if IA and IB are both non-zero.
TIX = 31.1, if IA <u>or</u> IB is non-zero.

(This page intentionally left blank)

## MOVE CHARACTER INSTRUCTION

The Move Character instruction moves 1-100 characters from one location in main memory to another.

## INSTRUCTION FIELDS

### Machine Operation Code

F---Binary 1000 (8).

### Address Specification

A---Address of the leftmost position of Operand-A.

B---Address of the leftmost position of Operand-B.

### Indexing Specification

IA--Index register for determining effective address of Operand-A.

IB--Index register for determining effective address of Operand-B.

### Common Partition Specification

AC--If AC is 0, A is an address in controlling partition.
If AC is 1, A is an address in Common.

BC--If BC is 0, B is an address in controlling partition.
If BC is 1, B is an address in Common.

### Length Specification

LA--Tens position of length of both Operand-A and Operand-B.

LB--Units position of length of both Operand-A and Operand-B.

## OPERAND FIELDS

### Operand-A Address

If IA is 0, then A is the effective address.
If IA is 1, 2, or 3, the corresponding index register is added to A to determine the effective address of Operand-A.

If AC is 1, the effective address lies in Common.

### Operand-B Address

If IB is 0, then B is the effective address.
If IB is 1, 2, or 3, the corresponding index register is added to B to determine the effective address of Operand-B.

If BC is 1, the effective address lies in Common.

### Operand Lengths

Operand-A and Operand-B are equal in length.
10LA + LB  Length of operands for Move Character instruction.
If 10LA + LB  00, 100 is the length of the operands.

## OPERATION

### General Description

Operand-A is copied into Operand-B, one position at a time, from left to right, starting with the leftmost position of Operand-A and writing it into the leftmost position of Operand-B.

### Condition Code

2, after completion of the Move Character instruction.

### Execution Time (T) in Microseconds

T = 40.0 + 11.1(10LA + LB) + TIX

| | | |
|---|---|---|
| Key: | TIX = 0.0 | if IA and IB are both zero |
| | TIX = 58.9 | if IA and IB are both non-zero |
| | TIX = 31.1 | if IA or IB is non-zero |

## PROGRAMMING HINTS

### Move Character VS Move Numeric

The Move Character instruction is similar to the Move Numeric instruction. The Move Numeric instruction will extract and copy only the numeric portion of a character (leaving the zone bits unchanged); the Move Character instruction will copy an entire character including both numeric and zone portions.

### Overlapping Operands

If Operand-A and Operand-B do not overlap, then Operand-A is unchanged by the Move Character instruction.

To shift the Operand-A data field one or more positions to the left (to a lower machine address) the Move Character instruction can be used when the operands overlap if the Operand-B address is not greater than the Operand-A address. Only the unlapped positions of Operand-A will be unchanged.

To propagate a given character throughout a data field, put the character into the leftmost position of the field, and use the Move Character instruction as follows:

Operand-A address is the address of the data field.
Operand-B address is the address of the data field + 1.
Operand length must be 1 less than the data field length.

(This page intentionally left blank)

## MOVE NUMERIC INSTRUCTION

The Move Numeric instruction moves the numeric portion of 1-100 characters from one location in main memory to another. The zone bits of both fields are unchanged.

## INSTRUCTION FIELDS

### Machine Operation Code

F---Binary 1001 (9).

### Address Specification

A---Address of the leftmost position of Operand-A.

B---Address of the leftmost position of Operand-B.

### Indexing Specification

IA--Index register for determining effective address of Operand-A.

IB--Index register for determining effective address of Operand-B.

### Common Partition Specification

AC--If AC is 0, A is an address in controlling partition. If AC is 1, A is an address in Common.

BC--If BC is 0, B is an address in controlling partition. If BC is 1, B is an address in Common.

### Length Specification

LA--Tens position of length of both Operand-A and Operand-B.

LB--Units position of length of both Operand-A and Operand-B.

## OPERAND FIELDS

### Operand-A Address

If IA is 0, then A is the effective address.
If IA is 1, 2, or 3, the corresponding index register is added to A to determine the effective address of Operand-A.

If AC is 1, the effective address lies in common.

### Operand-B Address

If IB is 0, then B is the effective address.
If IB is 1, 2, or 3, the corresponding index register is added to B to determine the effective address of Operand-B.

If BC is 1, the effective address lies in common.

### Operand Lengths

Operand-A and Operand-B are equal in length.
10LA + LB = Length of operands for Move Numeric instruction.
If 10LA + LB = 00, 100 is the length of the operands.

## OPERATION

### General Description

The numeric portion of Operand-A is copied into the numeric portion of Operand-B, one position at a time, from left to right, starting with the leftmost position of Operand-A and writing it into the leftmost position of Operand-B.

### Condition Code

2, after completion of the Move Numeric instruction.

### Execution Time (T) in Microseconds

T 40.0 + 11.1(10LA + LB) + TIX

| | | |
|---|---|---|
| Key: | TIX = 0.0 | ifIA and IB are both zero |
| | TIX = 58.9 | if IA and IB are both non-zero |
| | TIX = 31.1 | if IA <u>or</u> IB is non-zero. |

## PROGRAMMING HINTS

### Move Numeric VS Move Character

The Move Numeric instruction is similar to the Move Character instruction. The Move Numeric instruction will extract and copy only the numeric portion of a character (leaving the zone bits unchanged); the Move Character instruction will copy an entire character including both numeric and zone portions.

### Overlapping Operands

If Operand-A and Operand-B do not overlap, then Operand-A is unchanged by the Move Numeric instruction.

To shift the Operand-A numeric field one or more positions to the left (to a lower machine address) the Move Numeric instruction can be used when the operands over-lap if the Operand-B address is not greater than the Operand-A address. The unlapped positions of Operand-A and all zone bits in both operands will be unchanged.

To propagate a given digit throughout a data field, put the digit into the leftmost position of the field, and use the Move Numeric instruction as follows:

> Operand-A address is the address of the data field.
> Operand-B address is the address of the data field + 1.
> Operand length must be 1 less than the data field length.

The Move Numeric instruction enables the programmer to change the numeric portions of instructions. If is most frequently used in address modification (A and B fields). It is also useful in varying the LA and/or LB fields.

(This page intentionally left blank)

## MULTIPLY INSTRUCTION

The Multiply instruction computes the algebraic product of two 1 to 10 position numeric operands.

## INSTRUCTION FIELDS

### Machine Operation Code

F---Binary 0110 (6).

### Address Specification

A---Address of the leftmost position of Operand-A.

B---Address of the leftmost position of Operand-B, and Address of the leftmost position of Product field.

### Indexing Specification

IA--Index register for determining effective address of Operand-A.

IB--Index register for determining effective address of Operand-B.

### Common Partition Specification

AC--If AC is 0, A is an address in controlling partition.
If AC is 1, A is an address in Common.

BC--If BC is 0, B is an address in controlling partition.
If BC is 1, B is an address in Common.

### Length Specification

LA--Length of Operand-A.

LB--Length of Operand-B.

LB + LA--Length of Product field.

## OPERAND FIELDS

### Operand-A Address

If IA is 0, then A is the effective address.
If IA is 1, 2, or 3, the corresponding index register is added to A to determine the effective address of Operand-A.

If AC is 1, the effective address lies in Common.

### Operand-B Address

If IB is 0, then B is the effective address.
If IB is 1, 2, or 3, the corresponding index register is added to B to determine the effective address of Operand-B.

If BC is 1, the effective address lies in Common.

### Operand Lengths

If LA is 0, the length of Operand-A is 10 characters.
If LA is 1 thru 9, the length of Operand-A is 1 thru 9 characters.

If LB is 0, the length of Operand-B is 10 characters.
If LB is 1 thru 9, the length of Operand-B is 1 thru 9 characters.

## PRODUCT FIELD

The product field may be thought of as the multiplier field (Operand-B) with a rightward extension of a length equal to that of the multiplicand (Operand-A); hence, the product field will be located at the Operand-B address and will have the length LB + LA.

## OPERATION

### General Description

Operand-A is the multiplicand.

Operand-B is the multiplier.

The product is developed in the extended Operand-B field. The extension is cleared to zeros before the following computation is begun:

1---The rightmost digit of the multiplier is put into a register to govern the number of times the multiplicand will be added into the rightmost positions of the product field.

2---The rightmost position of the multiplier field is cleared to provide an extra left position for the add operation.

3---The multiplicand is added into the rightmost positions of the product field the number of times specified by the governing multiplier digit stored in the register.

4---Steps 1, 2, and 3 are repeated with the next left digit of the multiplier acting as governing digit. The multiplicand is repeatedly added into the next left positions of the product field. The process continues until the leftmost multiplier digit has served as governing digit.

Bit-7 is set OFF in all positions of the product except the rightmost position which is set to the sign of the product.

Bit-7 ON = factor signs differed.

Bit-7 OFF = factor signs were alike.

Bit-5 is set ON in all positions of the product field.

Operand-A is unchanged by the multiply operation if the fields do not overlap.

An overflow condition will never occur if the numeric portions of the numeric values are 0 thru 9.

## Condition Codes

After completion of the Multiply instruction:

1 = Negative, non-zero product.
2 = Zero product.
3 = Positive, non-zero product.

### Execution Time (T) in Microseconds

T 47.8 + 6.67 (LA )+ 10.0 (LB) + ((10.0 + 11.1 LA) (S)) + TIX.

Key:

| | |
|---|---|
| TIX = 0.0, | if IA and IB are both zero. |
| TIX =58.9, | if IA and IB are both non-zero. |
| TIX =31.1, | if IA <u>or</u> IB is non-zero. |

S = Sum of digits in Operand-B.

## PROGRAMMING HINTS

### Overlapped Operands

In case of overlapped operands, the result is unspecified.

### Overflow

Overflow will never occur if all characters in the numeric portions of the operands are the digits 0 thru 9. Overflow can occur if the numeric portions of the operands contain the following digits:

| | | |
|---|---|---|
| binary | 1010 | (10) |
| | 1011 | (11) |
| | 1100 | (12) |
| | 1101 | (13) |
| | 1110 | (14) |
| | 1111 | (15) |

## READ INSTRUCTION

The Read instruction moves data from an input device to sequential locations in Main Memory.

## INSTRUCTION FIELDS

### Machine Operation Code

F---Binary 0000 (0).

### Channel Specification

LB--If bit-1 is 0, reading will be routed through the FAC.
If bit-1 is 1, reading will be routed through the IOC.

### Mode Specification

LB--If bit-4 is 0, reading will be in the "fill" mode.
If bit-4 is 1, reading will be in the "non-fill" mode.

### Input Device Specification

LA--Device address 0 - 9 for IOC.
Device address 0 - 4 and 8 for FAC.

### Input Address Specification

A---Address of input area.

B---If the input device is not the disc, B is the count.
If the input device is the disc, B is the indirect disc address.

The indirect disc address points to a 6-character field which contains the disc address. The format of this field is illustrated in Figure 3.

### Indexing Specification

IA--Index register for determining effective address of input area.

IB--Index register for determining effective indirect disc address <u>or</u> effective count.

### Common Partition Specification

AC--If AC is 0, A is an address in controlling partition.
If AC is 1, A is an address in Common.

BC--If B is a count, the BC is ignored.

If BC is 0, B is an address in controlling partition.
If BC is 1, B is an address in Common.

### Count Specification

If the disc is the input device, the count is always 100 and is not specified in the Read instruction.

If the input device is not the disc, B is the count. A count of 0000 is interpreted as 10,000.

## OPERATION

### IOC General Operation

A Read instruction that specifies data transmission through the IOC is executed incrementally. The instruction is first decoded, and parameters are set into registers A, B, and P for the partition initiating the operation. A signal is sent to the IOC to alert the input device. Control is then relinquished to the next partition. The fulfillment of the Read instruction is performed between the execution of instructions in the other partitions. Before each instruction begins, the CPU stores one character for each IOC that has a character ready. This incremental operation proceeds as follows:

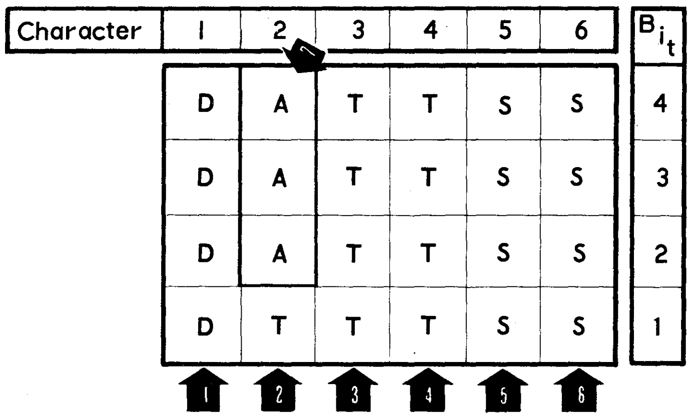
1---An IOC requests a character from the input device.

2---The input device gives a character to the IOC which sets a signal to inform the CPU of "character ready".

3---Between instruction executions, the CPU discovers the signal, stores the character being held by the IOC, and updates the parameter registers.

4---If the number of characters already transmitted has reached the count specified in the Read instruction, no more characters are requested. If the count has not been reached, steps 1, 2, 3, and 4 are repeated.

Only the numeric portions (1-4) of each character are used for specifying this information. Bit 7 may be either 0 or 1; Bit 5 must always be 1. The information is specified as follows.



1 DEVICE NUMBER (0-9)

2 HUNDREDS DIGIT (0 or 1) OF A THREE DIGIT TRACK NUMBER

3 TENS DIGIT (0-9) OF A THREE DIGIT TRACK NUMBER

4 UNITS DIGIT (0-9) OF A THREE DIGIT TRACK NUMBER

5 TENS DIGIT (0-9) OF A TWO DIGIT SECTOR NUMBER

6 UNITS DIGIT (0-9) OF A TWO DIGIT SECTOR NUMBER

7 ARM NUMBER (0-4)

NOTE:

- The bits in characters 1,3,4,5, and 6 have the following values:

    Bit 1 has the value 1 when it is ON.
    Bit 2 has the value 2 when it is ON.
    Bit 3 has the value 4 when it is ON.
    Bit 4 has the value 8 when it is ON.

- The bits in character 2 have the following values:

    Bit 1 has the value 1 when it is ON.
    Bit 2 has the value 1 when it is ON.
    Bit 3 has the value 2 when it is ON.
    Bit 4 has the value 4 when it is ON.

Figure 3 Disc Address Matrix Format

If control returns to the partition which initiated the Read instruction before the count is satisfied, control simply passes to the next partition. If the count is satisfied when control returns to the partition which initiated the Read instruction, a Condition Code is set (see description of individual devices), the execution continues with the next sequential instruction following the Read instruction.

## FAC General Operation

A Read instruction that specifies data transmission through the FAC does not relinquish control to the next partition during data transmission. Instead, the CPU is devoted exclusively to storing data provided by the FAC until the entire count is satisfied. During this period the CPU does not service any IOC. Service to the IOCs resumes at the completion of the Read instruction.

## Disc Access Sequence

A Read instruction addressing the disc does not typically pre-empt the CPU (as described above) immediately. It is sometimes necessary to wait until the disc is free, and then to wait while the heads move to the required cylinder. During either type of wait, control passes to the neighboring partition, and returns again in normal sequence.

A disc is <u>free</u> if it is not bound to another partition. It is <u>bound</u> to a given partition as soon as the partition institutes a seek upon it; it remains bound until data transmission is complete.

If the disc is bound to another partition when a Read instruction is attempted, control merely passes to the next partition. The Read instruction will be attempted again when control returns to the host partition.

If the disc is free when a Read instruction is attempted, a seek is automatically instituted, and the disc is then bound to the host partition. If head movement is necessary, control passes to the next partition. Transmission begins when the heads reach the proper cylinder, when control returns to the host partition, and when the desired sector rotates into place.

If the heads are already "on cylinder" when the seek is instituted, control remains with the host partition. Transmission begins as soon as the desired sector rotates into place.

When the disc record is entirely transmitted, a Condition Code is set to indicate the outcome. The CPU services any outstanding IOC for signals, and execution continues with the next sequential instruction following the Read instruction.

Succeeding instructions in the host partition which access the same cylinder will be executed without switching partitions. The first attempt to access another cylinder, however, will free the disc and pass control to the next partition. When control again returns to the host partition, the Read/Write instruction will be subject to the entire wait process (as described above).

## Fill and Non-Fill

A Read instruction using the IOC will terminate prematurely if the input device sends the IOC a Unit Separator character. In such a case, the Unit Separator character is not stored. Remaining positions of the input area are normally filled with blank characters. If the non-fill option was requested (bit-4 of instruction field LB), the remaining positions in the input area are left undisturbed.

## Condition Codes

After completion of the Read instruction.

1 = Error
2 = Normal
3 = Flag
4 = Fault

## Execution Time (T) in Microseconds

T - 91.1 + TIX for an Input/Output Channel (IOC).

T = 73.3 + TIX for a File Access Channel (FAC).

| | |
|---|---|
| Key: | TIX = 0.0, if IA and IB are both zero. |
| | TIX = 58.9, if IA and IB are both non-zero. |
| | TIX = 31.1, if IA <u>or</u> IB is non-zero. |

(This page intentionally left blank)

## SUBTRACT INSTRUCTION

The Subtract instruction computes the algebraic difference between the numeric portions of the two operands. The difference replaces the second operand (the minuend) and leaves the first operand unchanged if the fields do not overlap.

## INSTRUCTION FIELDS

### Machine Operation Code

F---Binary 0111 (7).

### Address Specification

A---Address of the leftmost position of Operand-A.

B---Address of the leftmost position of Operand-B.

### Indexing Specification

IA--Index register for determining effective address of Operand-A.

IB--Index register for determining effective address of Operand-B.

### Common Partition Specification

AC--If AC is 0, A is address in controlling partition.
If AC is 1, A is address in Common.

BC--If BC is 0, B is address in controlling partition.
If BC is 1, B is address in Common.

### Length Specification

LA--Length of Operand-A.

LB--Length of Operand-B.

## OPERAND FIELDS

### Operand-A Address

If IA is 0, then A is the effective address.
If IA is 1, 2, or 3, the corresponding index register is added to A to determine the effective address of Operand-A.

If AC is 1, the effective address lies in Common.

### Operand-B Address

If IB is 0, then B is the effective address.
If IB is 1, 2, or 3, the corresponding index register is added to B to determine the effective address of Operand-B.

If BC is 1, the effective address lies in Common.

### Operand Lengths

If LA is 0, the length of Operand-A is 10 characters.
If LA is 1 thru 9, the length of Operand-A is 1 thru 9 characters.

If LB is 0, the length of Operand-B is 10 characters.
If LB is 1 thru 9, the length of Operand-B is 1 thru 9 characters.

## OPERATION

### General Description

The subtract operation proceeds from right to left starting with the rightmost character of Operand-A and Operand-B. Character by character, the algebraic difference is developed in Operand-B.

The hardware acts as though the sign of Operand-A were reversed. In every other respect the instruction behaves like the Add instruction.

If Operand-A is shorter than Operand-B, the operation proceeds normally until Operand-A is exhausted. After that, the process continues in similar fashion except that a zero character is automatically substituted every time the logic calls for a character from Operand-A. In effect, Operand-A is given enough preceding zeros to make it the same length as Operand-B.

If Operand-A is longer than Operand-B, subtraction stops after the leftmost position in Operand-B has been subtracted. The remaining positions in Operand-A are ignored, and do not affect the difference or the Condition Code.

The algebraic sign of the difference is placed in bit-7 of the rightmost position of Operand-B, and bit-5 is turned on. Except for the rightmost character, the other zone bits of Operand-B are unchanged. Operand-A is unchanged by the subtract operation.

If the difference exceeds the capacity of Operand-B, a carry-to-the-left from the leftmost position does not occur. Condition Code 4 is set to indicate the overflow.

### Condition Codes

After completion of the Subtract instruction.

1 = Negative, non-zero difference.
2 = Zero difference.
3 = Positive, non-zero difference.
4 = Overflow.

### Execution Time (T) in Microseconds

T = 42.2 + 3.3 (LA) + 10.0 (LB) + TIX + TOD, if LA is equal to or less than LB.

T = 42.2 + 11 (LA) + 12.2 (LB) + TIX + TOD, if LA is greater than LB.

| | |
|---|---|
| Key: | TIX = 0.0, if IA and IB are both zero. |
| | TIX = 58.9, if IA and IB are both non-zero. |
| | TIX = 31.1, if IA <u>or</u> IB is non-zero. |
| | TOD = 0.0, if an overdraft <u>does not</u> occur. |
| | TOD = 10.0 (LB), if an overdraft occurs. |
| | An overdraft will always occur when the absolute value of Operand-A exceeds the absolute value of Operand-B and they have like signs. |

## PROGRAMMING HINTS

### Overlapped Operands

In case of overlapped operands, the result is unspecified.

## WRITE INSTRUCTION

The Write instruction transmits data from sequential locations in Main Memory to an output device. A control option enables the Write instruction to communicate control information to the input or output device.

## INSTRUCTION FIELDS

### Machine Operation Code

F---Binary 0001 (1).

### Channel Specification

LB--If bit-1 is 0, writing will be routed through the FAC.
    If bit-1 is 1, writing will be routed through the IOC.

### Write Control Specification

LB--If bit-2 is 0, normal write.
    If bit-2 is 1, write control.

### Output Device Specification

LA--Device address 0 - 9 for IOC.
    Device address 0 - 4 and 8 for FAC.

### Output Address Specification

A---Address of output area.

B---If the output device is not the disc, B is the count. If the output device is the disc, B is the indirect disc address.

The indirect disc address points to a 6-character field which contains the disc address. The format of this field is illustrated in Figure 4.

### Indexing Specification

IA--Index register for determining effective address of output area.

IB--Index register for determining effective indirect disc address <u>or</u> effective count.

### Common Partition Specification

AC--If AC is 0, A is an address in controlling partition.
If AC is 1, A is an address in Common.

BC--If B is a count, BC is ignored.

If BC is 0, B is an address in controlling partition.
If BC is 1, B is an address in Common.

### Count Specification

If the disc is the output device, the count is always 100 and is not specified in the Write instruction.

If the output device is not the disc, B is the count. A count of 0000 is interpreted as 10,000.

## OPERATION

### IOC General Operation

A Write instruction that specifies data transmission through the IOC is executed incrementally. The instruction is first decoded, and parameters are set into registers A, B, and P for the partition initiating the operation. A signal is sent to the IOC to alert the output device. Control is then relinquished to the next partition. The transmission of characters occurs between the execution of instructions in the other partitions. Before each instruction begins, the CPU sends one character to each IOC which is ready to accept one. This incremental operation proceeds as follows:

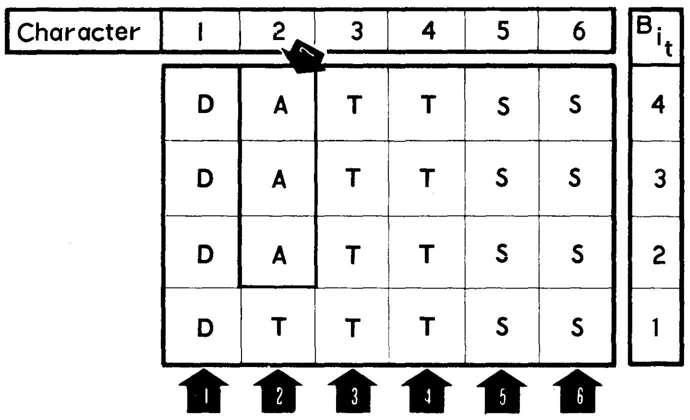
1---The IOC sets a signal to inform the CPU that it is ready to accept a character from the output area.

2---Between instruction executions, the CPU discovers the signal and checks the count balance. If the count has been reached, no more characters are sent to the IOC. If the count has not been reached, steps 3, 4, 1, and 2 are repeated, in that order.

3---The CPU gives a character to the IOC and updates the parameter registers.

4---As soon as it can, the output device accepts the character.

Only the numeric portions (1-4) of each character are used for specifying this information. Bit 7 may be either 0 or 1; Bit 5 must always be 1. The information is specified as follows.



1 DEVICE NUMBER (0-9)

2 HUNDREDS DIGIT (0 or 1) OF A THREE DIGIT TRACK NUMBER

3 TENS DIGIT (0-9) OF A THREE DIGIT TRACK NUMBER

4 UNITS DIGIT (0-9) OF A THREE DIGIT TRACK NUMBER

5 TENS DIGIT (0-9) OF A TWO DIGIT SECTOR NUMBER

6 UNITS DIGIT (0-9) OF A TWO DIGIT SECTOR NUMBER

7 ARM NUMBER (0-4)

NOTE:

- The bits in characters 1,3,4,5, and 6 have the following values:

  Bit 1 has the value 1 when it is ON.
  Bit 2 has the value 2 when it is ON.
  Bit 3 has the value 4 when it is ON.
  Bit 4 has the value 8 when it is ON.

- The bits in character 2 have the following values:

  Bit 1 has the value 1 when it is ON.
  Bit 2 has the value 1 when it is ON.
  Bit 3 has the value 2 when it is ON.
  Bit 4 has the value 4 when it is ON.

Figure 4 Disc Address Matrix Format

If control returns to the partition which initiated the Write instruction before the count is satisfied, control simply passes to the next partition. If the count is satisfied when control returns to the partition which initiated the Write instruction, a Condition Code is set (see description of individual devices), and execution continues with the next sequential instruction following the Write instruction.

### FAC General Operation

A Write instruction that specifies data transmission through the FAC does not relinquish control to the next partition during data transmission. Instead, the CPU is devoted exclusively to feeding data to the FAC until the entire count is satisfied. During this period the CPU does not service any IOC. Service to the IOCs resumes at the completion of the Write instruction.

### Disc Access Sequence

A Write instruction addressing the disc does not typically pre-empt the CPU (as described above) immediately. It is sometimes necessary to wait until the disc is free, and then to wait while the heads move to the required cylinder. During either type of wait, control passes to the neighboring partition, and returns again in normal sequence.

A disc is <u>free</u> if it is not bound to another partition. It is <u>bound</u> to a given partition as soon as the partition institutes a seek upon it; it remains bound until data transmission is complete.

If the disc is bound to another partition when a Write instruction is attempted, control merely passes to the next partition. The Write instruction will be attempted again when control returns to the host partition.

If the disc is free when a Write instruction is attempted, a seek is automatically instituted, and the disc becomes bound to the host partition. If head movement is necessary, control passes to the next partition. Transmission begins when the heads reach the proper cylinder, when control returns to the host partition, and when the desired sector rotates into place.

If the heads are already "on cylinder" when the seek is instituted, control remains with the host partition. Transmission begins as soon as the desired sector rotates into place.

When the disc record is entirely transmitted, a Condition Code is set to indicate the outcome. The CPU services any outstanding IOC signals, and execution continues with the next sequential instruction following the Write instruction.

Succeeding instructions in the host partition which access the same cylinder will be executed without switching partitions. The first attempt to access another cylinder, however, will free the disc and pass control to the next partition. When control again returns to the host partition, the Write instruction will be subject to the entire wait process (as described above).

## Write Control Mode

A Write instruction may specify the transmission of control characters to the external input/output device by having bit-2 of the LB instruction field ON. The information in the output area is sent to the external device one character at a time and exerts a controlling effect. The particular effect depends upon the information transmitted and upon the external device. As soon as the last character is accepted by the external device, program execution is free to continue even though the controlling effect is not yet realized. On the opposite page is a table showing how each internal character is converted to external form by an IOC Write Control instruction.

## Condition Codes

After completion of the Write instruction.

1 = Error
2 = Normal
3 = Flag
4 = Fault

## Execution Time (T) in Microseconds

T = 91.1 + TIX for an Input/Output Channel (IOC).

T = 73.3 + TIX for a File Access Channel (FAC).

```
    Key:   TIX =  0.0, if IA and IB are both zero.
           TIX = 58.9, if IA and IB are both non-zero.
           TIX = 31.1, if IA or  IB is non-zero.
```

| | Same Characters | | Changed Characters | |
|---|---|---|---|---|
| | Internal | External | Internal | External |
| Bits 7 | 0 | 0 | 1 | 0 |
| 6 | | 1 | | 0 |
| | SP | SP | @ | NUL |
| | ! | ! | A | SOH |
| | " | " | B | STX |
| | # | # | C | ETX |
| | $ | $ | D | EOT |
| | % | % | E | ENQ |
| | & | & | F | ACK |
| | ' | ' | G | BEL |
| | ( | ( | H | BS |
| | ) | ) | I | HT |
| | * | * | J | LF |
| | + | + | K | VT |
| | , | , | L | FF |
| | - | - | M | CR |
| | . | . | N | SO |
| | / | / | O | SI |
| | 0 | 0 | P | DLE |
| | 1 | 1 | Q | DC1 |
| | 2 | 2 | R | DC2 |
| | 3 | 3 | S | DC3 |
| | 4 | 4 | T | DC4 |
| | 5 | 5 | U | NAK |
| | 6 | 6 | V | SYB |
| | 7 | 7 | W | ETB |
| | 8 | 8 | X | CAN |
| | 9 | 9 | Y | EM |
| | : | : | Z | SUB |
| | ; | ; | { | ESC |
| | < | < | \ | FS |
| | = | = | } | GS |
| | > | > | ^ | RS |
| | ? | ? | _ | US |

Table 4 Write Control Conversions

Touch & Know
Business Machines by
SINGER

Publication No. 42-2500-01