

REFERENCE MANUAL

ASSEMBLER 1
(Edition B)



SYSTEM TEN COMPUTER BY **SINGER**

SINGER
FRIDEN DIVISION

REFERENCE MANUAL

ASSEMBLER 1
(Edition B)



Publication No. 40-029-1
(Control No. B004PB)

November 1971

*A trademark of The Singer Company.

SINGER
FRIDEN DIVISION

2350 WASHINGTON AVE.
SAN LEANDRO, CALIF. 94577

PREFACE

This manual explains how to write programs in the Assembler I programming language, and how to assemble these programs. The reader should be familiar with the fundamental techniques of programming and the operating principles of System Ten. Experience with at least one other assembler or class work covering Assembler I is desirable. In addition, the System Ten Programmer's Machine Reference Manual is recommended as a prerequisite. Those using Assembler I in a disc environment should also read the Disc Management Facility User's Reference Manual.

TABLE OF CONTENTS

Section 1 INTRODUCTION

| | |
|--|------|
| MEMORY REQUIREMENTS | 1-2 |
| I/O DEVICE REQUIREMENT | 1-3 |
| Card Version | 1-3 |
| Disc Version | 1-5 |
| AUXILIARY SOFTWARE | 1-5 |
| STATEMENT REPERTOIRE | 1-6 |
| OVERVIEW OF ASSEMBLY PROCESS | 1-7 |
| LABEL TABLE | 1-9 |
| SOURCE DECK STRUCTURE | 1-9 |
| OVERLAY STRUCTURE | 1-10 |
| STATEMENT FORMAT | 1-12 |
| Character Set | 1-12 |
| * as an Address | 1-12 |
| SOURCE STATEMENT FIELDS | 1-14 |
| Label Field | 1-14 |
| Operation Field | 1-15 |
| Operand Field | 1-16 |
| Comments Fields | 1-16 |
| Program Identification Field | 1-16 |
| Sequence Number Field | 1-16 |
| STANDARD CODING FORMS & SOURCE CARDS | 1-17 |
| COMMENTS STATEMENT | 1-17 |
| DEBUG STATEMENT | 1-17 |
| INSTRUCTION ALIGNMENT | 1-19 |
| OPERATION FIELD KEYWORDS | 1-19 |
| SYNTACTICAL ERRORS | 1-19 |
| OPERAND FORMS | 1-20 |
| Operand Fields vs. Instruction Fields | 1-21 |
| Positional Aspects | 1-21 |
| Internal Blanks | 1-22 |
| LA, LB=10 | 1-22 |
| L=100 | 1-22 |
| Operand Constants and Symbols | 1-22 |
| Implied Operand Forms | 1-24 |
| INSTRUCTION FIELDS AND HOW THEY ARE GENERATED | 1-26 |
| F (Op Code) | 1-26 |
| A, B (Operand Address/Count) | 1-27 |
| AC, BC (Common/Partition) | 1-27 |
| IA, IB (Index Register Specification) | 1-28 |
| LA, LB (Length, Device, Channel, Variant) | 1-29 |
| ASSEMBLER COMMANDS | 1-29 |
| Listing Control Commands | 1-29 |
| Address Control Commands | 1-30 |
| Termination Control Commands | 1-30 |
| Memory Allocation Commands | 1-30 |

TABLE OF CONTENTS

Section 2 MACHINE INSTRUCTION STATEMENTS

| | |
|---|------|
| INTRODUCTION | 2-1 |
| ADD | 2-2 |
| BRANCH (Conditional, Unconditional, Switch) | 2-4 |
| BRANCH AND LINK | 2-7 |
| BRANCH ON SERVICE REQUEST | 2-9 |
| COMPARE | 2-10 |
| DIVIDE | 2-13 |
| EDIT | 2-15 |
| EXCHANGE | 2-19 |
| FORM NUMERIC | 2-21 |
| MOVE CHARACTER | 2-23 |
| MOVE NUMERIC | 2-24 |
| MULTIPLY | 2-25 |
| READ | 2-27 |
| SUBTRACT | 2-33 |
| WRITE | 2-35 |

Section 3 ASSEMBLER COMMANDS

| | |
|--------------------|------|
| INTRODUCTION | 3-1 |
| COMMON | 3-2 |
| DM | 3-4 |
| EJECT | 3-11 |
| END | 3-12 |
| EXEC | 3-13 |
| NORMAL | 3-14 |
| ORG | 3-16 |
| SPACE | 3-18 |
| TITLE | 3-19 |

Section 4 ASSEMBLER OUTPUT

| | |
|--------------------------------------|-----|
| INTRODUCTION | 4-1 |
| LISTING OUTPUT | 4-1 |
| Error Statement Listing | 4-2 |
| Label Table Listing | 4-5 |
| Program Listing | 4-6 |
| Symbol Cross-Reference Listing | 4-7 |
| ERROR MESSAGES | 4-8 |
| OBJECT DECK FORMATS | 4-9 |
| T Card | 4-9 |
| S Card | 4-9 |

Section 5 USING THE ASSEMBLER

| | |
|----------------------------------|------|
| ASSEMBLER I (CARD VERSION) | 5-1 |
| Fixed Parameters | 5-1 |
| Variable Parameters | 5-3 |
| Assembling Source Program | 5-4 |
| ASSEMBLER I (DISC VERSION) | 5-7 |
| Fixed Parameters | 5-7 |
| Installing the Assembler | 5-9 |
| Filing Source Deck | 5-10 |
| Assembling Source Program | 5-11 |

APPENDICES

| | | |
|------------|---------------------------------------|-----|
| Appendix A | ASSEMBLER DECK PARAMETERS | A-1 |
| Appendix B | MACHINE INSTRUCTION STATEMENTS | B-1 |
| Appendix C | ASSEMBLER COMMANDS | C-1 |
| Appendix D | CONDITION CODES | D-1 |
| Appendix E | I/O DEVICE CONTROL CHARACTERS | E-1 |
| Appendix F | I/O DEVICE CONDITION CODES | F-1 |
| Appendix G | STANDARD DEVICE NUMBERS | G-1 |
| Appendix H | DISC ADDRESS | H-1 |
| Appendix I | CONVERSION TABLES | I-1 |
| Appendix J | BOOTSTRAP CARD AND FOUR-CARD LOADER . | J-1 |

GLOSSARY

INDEX

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

| FIGURE | | PAGE |
|--------|--------------------------------------|------|
| 1-1 | TYPICAL ASSEMBLY CONFIGURATION | 1-4 |
| 1-2 | FLOW OF ASSEMBLY PROCESS | 1-8 |
| 1-3 | AN OVERLAY PROGRAM | 1-11 |
| 1-4 | USASCII AND HOLLERITH CODES | 1-13 |
| 1-5 | SYSTEM TEN SOURCE CARD | 1-18 |
| 1-6 | ASSEMBLER CODING FORM | 1-18 |
| 1-7 | OPERANDS WITH INTERNAL BLANKS | 1-23 |
| 1-8 | OPERAND CONSTANTS AND SYMBOLS | 1-25 |
| 2-1 | SYSTEM TEN CHARACTER SET | 2-12 |
| 3-1 | EXAMPLES OF DM STATEMENTS | 3-5 |
| 4-1 | ERROR-FREE ASSEMBLER OUTPUT | 4-3 |
| 4-2 | ERROR STATEMENT LISTING | 4-4 |
| 5-1 | ASSEMBLER I JOB DECK | 5-6 |

Section 1 INTRODUCTION

MEMORY REQUIREMENTS
I/O DEVICE REQUIREMENT
AUXILIARY SOFTWARE
STATEMENT REPERTOIRE
OVERVIEW OF ASSEMBLY PROCESS
LABEL TABLE
SOURCE DECK STRUCTURE
OVERLAY STRUCTURE
STATEMENT FORMAT
SOURCE STATEMENT FIELDS
STANDARD CODING FORMS & SOURCE CARDS
COMMENTS STATEMENT
DEBUG STATEMENT
INSTRUCTION ALIGNMENT
OPERATION FIELD KEYWORDS
SYNTACTICAL ERRORS
OPERAND FORMS
INSTRUCTION FIELDS AND HOW THEY ARE GENERATED
ASSEMBLER COMMANDS

INTRODUCTION

Assembler I translates symbolically-coded source programs into machine-language object programs that can be executed by System Ten. The assembler assigns storage for the machine-language instructions, computes absolute addresses from symbolic addresses, and performs other functions that aid in preparing the object program for loading. The customer may select either of two versions of the assembler:

*Assembler I (Card Version) for operating in a
card I/O environment.

*Assembler I (Disc Version) for operating in a
disc I/O environment.

The assembler-language coding (described in sections 2 and 3) is the same for both versions of the assembler; however, the listing output (section 4) and installation and operating procedures (section 5) differ for each version. Essentially, the card version is loaded into memory manually along with each user source program processed. However, the disc version is read onto disc storage once, through the Disc Management Facility (DMF), and is thereafter called from disc residence into memory through DMF each time a user source program is processed. With the disc version, the user source program must also be input from a DMF disc file. Throughout this manual, the distinction between the card and disc versions of the assembler will be stressed only when necessary to point out differences.

INTRODUCTION

MEMORY REQUIREMENTS

The assembler program functions within the partition into which it is loaded. The size of the partition and the amount of space allotted in common determine the number of labels the assembler can process.

The assembler requires at least a 9K partition. The highest available positions of the partition are used for the label table. In addition, the label table may overflow into whatever common region locations are allowed by the installation parameters in the assembler deck. (See Section 5 and Appendix A.)

The maximum number of labelled statements is as follows:

| | <u>9K Partition</u> | <u>10K Partition</u> |
|---------------|---------------------|----------------------|
| Card Version: | 158 | 235 |
| Disc Version: | 150 | 226 |

plus 77 labels for each 1001 positions of common core used.

I/O DEVICE REQUIREMENTS

Assembler I (Card Version)

With the card version of the assembler, an input device with load capability must be assigned as IOC device 0. This device is used to begin a load sequence that bootstraps a loader into the user's partition, where it remains throughout assembly. The following devices possess load capability:

Model 30 Card Reader

Model 70 Workstation

Model 7102 Communications Terminal

Various devices are required for source program input and listing and object program output. The source program can be input from the Model 30 Card Reader.

The listing output (error statement, label table, and program listings) can be output to any of the following:

Model 50 Line Printer

Model 70 Workstation

Model 7102 Communications Terminal

The object program output can be transmitted to the Model 35 Card Punch.

As a typical example (figure 1-1), an installation might have a Workstation connected as device 0 to function as a load device and a card reader assigned as device 1 to serve as a source input device. A line printer attached as device 2 might act as listing device and a card punch attached as device 4 might act as object device.

The particular configuration is flexible. The user may specify which device the assembler is to use for source input, for listings, and for object program output by using the Assembler Deck Parameters described in section 5. The IOC device number assignments recommended for standard use are presented in appendix G.

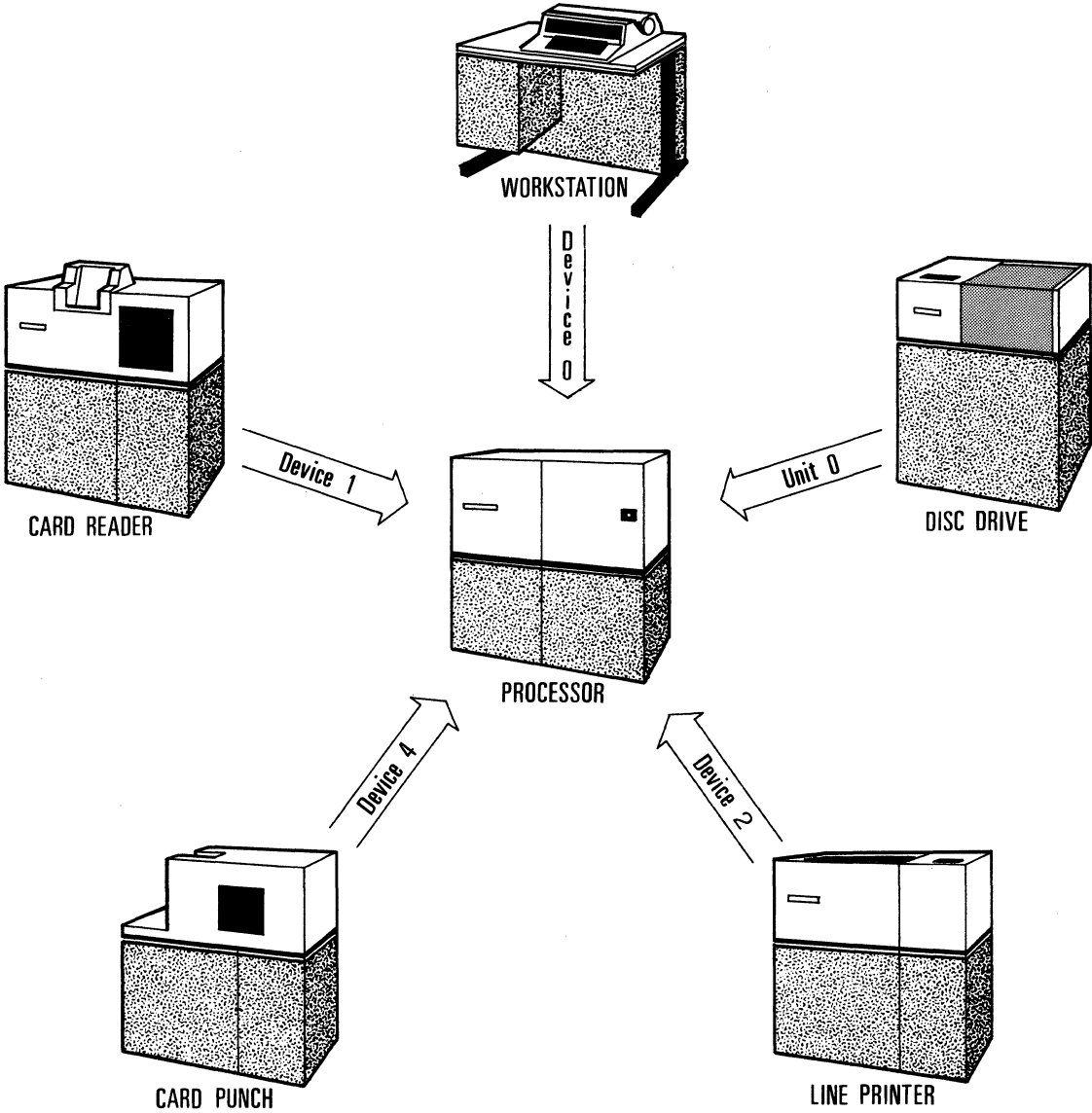


Figure 1-1. Typical Assembly Configuration

Assembler I (Disc Version)

With the disc version of the assembler, the Model 70 Workstation or a Model 7102 Communications Terminal is used to call the assembler from disc residence into memory through DMF. The source program must also be input from a disc file through DMF.

The disc version of Assembler I uses the permanent logical unit assignments from low core for determining the assembler command input, listing output, and object output devices. These assignments are established by the permanent mode variant of the DMF MAINT ASSIGN function.

The assembler parameter statements are read from the COMMAND logical unit (_LRDR), and can be assigned to the following:

Model 30 Card Reader
Model 70 Workstation
Model 80 Display
Model 7102 Communications Terminal

The listing output is transmitted to the PRINT logical unit (_LLST). This logical unit can be set to IGN (ignore) to suppress the listing. The following devices can be assigned as PRINT:

Model 50 Line Printer
Model 70 Workstation
Model 7102 Communications Terminal

The object program can be output to the Model 35 Card Punch or to a DMF file on a Model 40 Disc Drive. These assignments are controlled by the OBJECT= assembler command when directing the object program to disc; when OBJECT= is omitted, the assembler defaults to the PUNCH logical unit (which may be IGN).

AUXILIARY SOFTWARE

The card version of the assembler requires a loader capable of loading the assembler.

The disc version of the assembler is installed, loaded, and operated in conjunction with DMF.

In no other respect does the assembler depend upon other software.

INTRODUCTION

STATEMENT REPERTOIRE

For each of the 13 System Ten CPU instructions, there is a corresponding assembler machine instruction statement. The machine instruction statements are listed below and discussed in section 2.

| <u>INSTRUCTION NAME</u> | <u>OP CODE</u> |
|-------------------------|----------------|
| Add | A |
| Divide | D |
| Form Numeric | FN |
| Multiply | M |
| Subtract | S |
| Compare | C |
| Exchange | X |
| Edit | E |
| Move Character | MC |
| Move Numeric | MN |
| Read | R |
| Write | W |
| Branch | BC |

To control the assembly, there are nine assembler commands (discussed in section 3):

```
COMMON
END
EJECT
EXEC
NORMAL
ORG
SPACE
TITLE
DM
```

In addition, there are various assembler parameter statements, used to specify input/output options.

For the card version, there are three assembler parameter statements:

```
DEBUG=
COMTXT=
DATE=
```

For the disc version, there are six assembler parameter statements:

```
DEBUG=
COMTXT=
DATE=
XREF=
SOURCE=
OBJECT=
```

Unlike machine instruction statements and assembler commands, assembler parameter statements are not coded as part of the user's program. Instead, they are punched on a special parameter card in the assembler object deck (in the card version) or are entered separately via the Workstation, Communications Terminal, or Card Reader (for the disc version). Assembler parameter statements are discussed in section 5.

OVERVIEW OF ASSEMBLY PROCESS

The assembler is essentially a two-pass assembler. That is, it makes two passes over the source program. Pass one is preceded by an initialization phase which establishes assembly parameters according to user-supplied parameter statements. Pass two is preceded by a printout of the label table. The main steps in the process are shown in figure 1-2 and described below.

1. The loader loads the "Parameter Statement Interpreter" program and passes control to it.
2. The Parameter Statement Interpreter reads any user supplied parameter statements and sets parameters accordingly. A unit separator is necessary to return control to the loader.
3. The loader loads Pass One of the assembler program and passes control to it.
4. The Pass One program reads the source program, checks for errors (listing them), and builds the label table. The END card stops the reading and returns control to the loader.
5. The loader loads the Label Table Printer program and passes control to it.
6. The Label Table Printer program prints the label table and passes control to the loader.
7. The loader loads the Pass Two program and gives control to it.
8. The Pass Two program makes the second pass over the source deck if no errors were detected in Pass One. Pass Two is omitted if errors were detected during Pass One. The assembler generates a program listing and object deck. Again, the END card returns control to the loader.

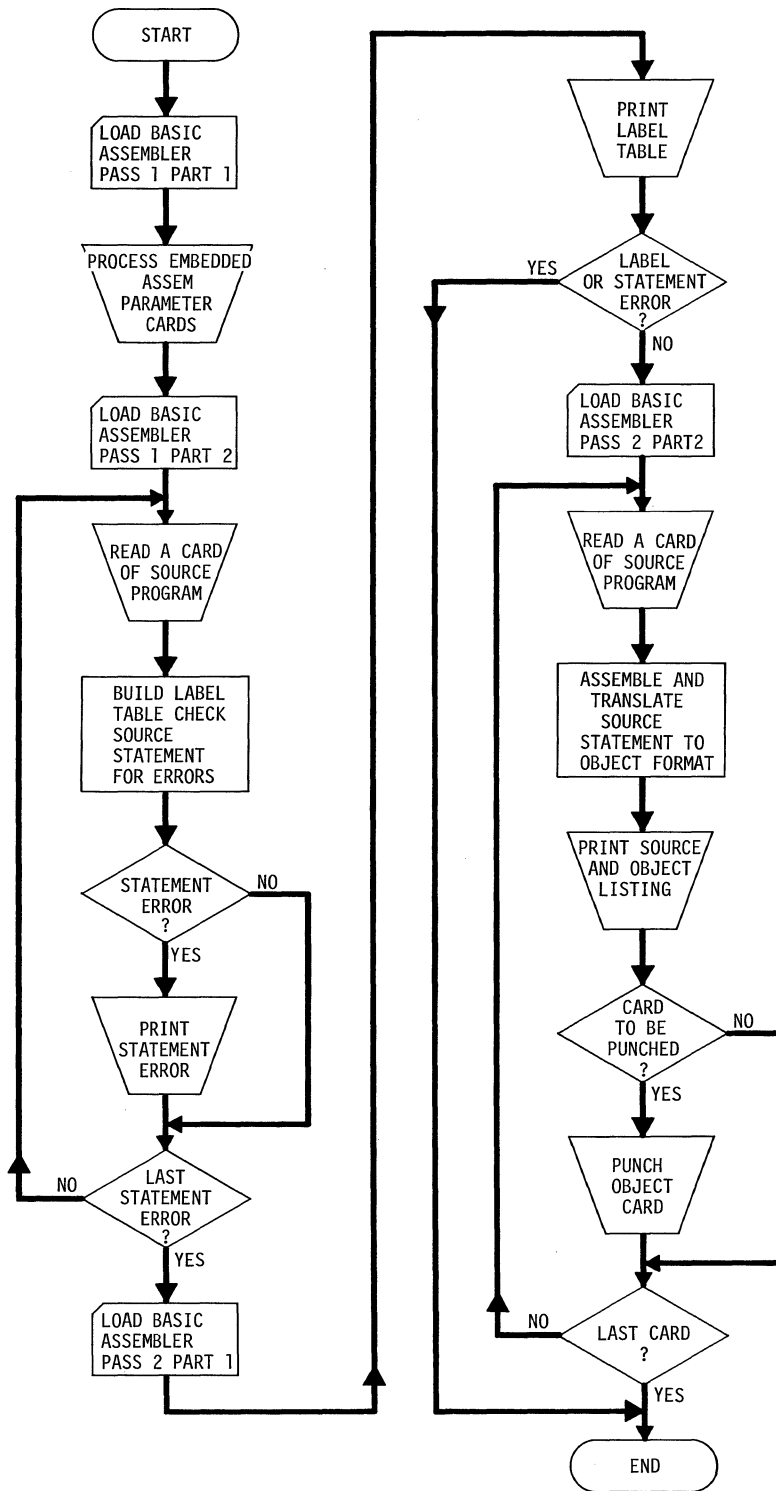


Figure 1-2. Flow of Assembly Process

LABEL TABLE

The assembler makes two passes over the source program.

In the first pass, each statement is analyzed for the amount of memory it preempts. Each new label enters the label table. For each label thus entered, the table shows the address assigned to the label by the assembler, an indication as to whether the address is in common or partition, an implied length associated with the label, and the implied index.

In the second pass, the assembler substitutes the assigned address and common/partition indicator for each use of the label in the operand field of instruction statements and produces the object deck and listing. If an instruction expects a length indication corresponding to a given symbolic address specification, and if the programmer omits the length specification from the statement, the assembler supplies the length from the label table. Similarly, if the instruction expects an index indication corresponding to the symbolic address specification, and if the programmer omits the index specification, the assembler supplies the index from the label table.

In the data definition statements (DM) the programmer can exercise direct control over the length and index indications entered into the label table for data fields. By exercising this control, he is free to omit length and index indications from the instructions which manipulate or otherwise use the data fields.

The programmer has no direct control over the length and index indications entered into the label table for labelled machine instruction statements (which always assume a length of 10 and index of 0).

SOURCE DECK STRUCTURE

A source deck should conform to the following rules of order:

The TITLE card, if used, must be the first card in the deck.

The END card must be the last card in the deck.

From the first card to the END card, every card will be treated as a statement. If it cannot be recognized as a proper statement, it will be treated as an error. Thus, a completely blank card will be treated as an error.

INTRODUCTION

OVERLAY STRUCTURE

For programs where the total storage required is greater than the amount of storage available, the programmer may find it best to use some type of overlay structure. For example, he may plan his program so that a portion is initially loaded into core and executed. When execution of the initial load is finished, control returns to the loader to load the first overlay into the area previously occupied by the initial load. When the first overlay is executed, the second overlay is loaded, etc.

A programmer may invent a variety of ways to use overlays. Typically, the loader is kept in core throughout execution (locations 50-0299). Return to the loader is accomplished by coding a Branch instruction (which passes control to the loader's resume loading entry point) at each point in the logical flow where loading is desired.

The coding for the initial load and for the overlays may each be thought of as subdecks of the entire source deck.

The first card of each subdeck is usually an ORG command to reset the memory assignment counter and thus establish the proper load address for the coding which follows.

The last card of each subdeck is an EXEC command. The EXEC command causes the assembler to generate an S card in the object deck containing the start address for the overlay. When the loader encounters the S card, it stops loading and passes control to the start address.

The last subdeck must be followed by an END card to terminate the assembly.

Figure 1-3 is an example of the program listing generated by an overlay program.

Further directions for using overlays with a disc-oriented system are presented in the Disc Management Facility (DMF) User's Reference Manual.

| PAGE 0001 10/12/71 (4-02) PROGRAM LISTING FOR SYSTEM 10 EXECUTABLE OVERLAY PROGRAM | | | | | | | | | | | | |
|--|------|------------|----|------|---|---|------|---|---|------|--------|-----------------------------------|
| SEQ# | LOCN | INSTR/DATA | OP | A/R | L | I | B/S | L | I | LINE | LABEL | UPCODE OPERAND(S) AND/OR COMMENTS |
| 002 | 0000 | | | | | | 0300 | | | 0002 | ORG | 300 |
| 003 | 0300 | | | | | | 0000 | | | 0003 | TIMECD | DM OC80 |
| 004 | 0300 | | | | | | 0001 | | | 0004 | HOURS | DM C2 |
| 005 | 0302 | | | | | | 0001 | | | 0005 | RATE | DM N5(3) |
| 006 | 0307 | | | | | | 0380 | | | 0006 | ORG | 380 |
| 007 | 0380 | | | | | | 0080 | | | 0007 | PRINT | DM 80C1' ' |
| 008 | 0460 | | | | | | 0400 | | | 0008 | ORG | 400 |
| 009 | 0400 | | | | | | 0001 | | | 0002 | HOURSX | DM C2 |
| 010 | 0402 | | | | | | 0001 | | | 0010 | RATEX | DM C10 |
| 011 | 0412 | | | | | | 0001 | | | 0010 | AMOUNT | DM C10 |
| 012 | 0422 | | | | | | 0460 | | | 0012 | ORG | 460 |
| 013 | 0460 | | | | | | 0001 | | | 0005 | EDIT1 | DM C' . -' |
| 014 | 0465 | | | | | | 0001 | | | 0007 | EDIT2 | DM C' . -' |
| 015 | 0472 | | | | | | 0001 | | | 0007 | WORK1 | DM C7 |
| 016 | 0480 | 0030010080 | 00 | 0300 | 0 | 0 | 0080 | 1 | 0 | 0016 | START | R TIMECD(0),80(1) |
| 017 | 0490 | S0VP000000 | 11 | 0600 | 3 | 0 | 0000 | 0 | 0 | 0017 | BC | FINIS(3) |
| 018 | 0500 | P030020400 | 08 | 0300 | 0 | 0 | 0400 | 2 | 0 | 0018 | MC | HOURS,HOURSX |
| 019 | 0510 | P046050473 | 08 | 0460 | 0 | 0 | 0473 | 5 | 0 | 0019 | MC | EDIT1,WORK1+1 |
| 020 | 0520 | PP30230473 | 12 | 0302 | 0 | 0 | 0473 | 3 | 0 | 0020 | E | RATE,WORK1+1 |
| 021 | 0530 | P047270406 | 08 | 0472 | 0 | 0 | 0406 | 7 | 0 | 0021 | MC | WORK1,RATEX+4 |
| 022 | 0540 | 2PS0030302 | 06 | 0300 | 2 | 0 | 0302 | 3 | 0 | 0022 | M | HOURS,RATE |
| 023 | 0550 | P046570472 | 08 | 0465 | 0 | 0 | 0472 | 7 | 0 | 0023 | MC | EDIT2,WORK1 |
| 024 | 0560 | PP30250472 | 12 | 0302 | 0 | 0 | 0472 | 5 | 0 | 0024 | E | RATE(5),WORK1 |
| 025 | 0570 | P047270415 | 08 | 0472 | 0 | 0 | 0415 | 7 | 0 | 0025 | MC | WORK1,AMOUNT+3 |
| 026 | 0580 | 203X010080 | 01 | 0380 | 2 | 0 | 0080 | 1 | 0 | 0026 | W | PRINT(2),80(1) |
| 027 | 0590 | U0TX000000 | 11 | 0480 | 5 | 0 | 0000 | 0 | 0 | 0027 | BC | START(5) |
| 028 | 0600 | X0PV000000 | 11 | 0060 | 8 | 0 | 0000 | 0 | 0 | 0028 | FINIS | BC 60(8) |
| 029 | 0610 | | | | | | 0480 | | | 0029 | EXEC | START |
| 030 | 0610 | | | | | | 0480 | | | 0030 | ORG | START |
| 031 | 0480 | THE OVERLA | | | | | 0001 | | | 0023 | INF01 | DM C'THE OVERLAY HAS OCCURED' |
| 032 | 0503 | REPRINT TH | | | | | 0001 | | | 0021 | INF02 | DM C'REPRINT THE LAST LINE' |
| 034 | 0530 | 204X010023 | 01 | 0480 | 2 | 0 | 0023 | 1 | 0 | 0033 | AGAIN | W INF01(2),23(1) |
| 035 | 0540 | 205P310021 | 01 | 0503 | 2 | 0 | 0021 | 1 | 0 | 0034 | W | INF02(2),21(1) |
| 036 | 0550 | 203X010080 | 01 | 0380 | 2 | 0 | 0080 | 1 | 0 | 0035 | W | PRINT(2),80(1) |
| 037 | 0560 | X0UV000000 | 11 | 0560 | 8 | 0 | 0000 | 0 | 0 | 0036 | FINIS2 | BC FINIS2(8) |
| 038 | 0570 | | | | | | 0530 | | | 0037 | EXEC | AGAIN |
| 039 | 0570 | | | | | | | | | 0038 | END | |

| PAGE 0001 10/12/71 (4-02) | | | | LABEL TABLE FOR SYSTEM 10 EXECUTABLE OVERLAY PROGRAM | | | |
|---------------------------|------|------|------|--|--|--|--|
| LABEL | LOC | LNTH | INDX | MESSAGES | | | |
| AGAIN | 0530 | 10 | 0 | | | | |
| AMOUNT | 0412 | 10 | 0 | | | | |
| EDIT1 | 0460 | 05 | 0 | | | | |
| EDIT2 | 0465 | 07 | 0 | | | | |
| FINIS | 0600 | 10 | 0 | | | | |
| FINIS2 | 0560 | 10 | 0 | | | | |
| HOURS | 0300 | 02 | 0 | | | | |
| HOURSX | 0400 | 02 | 0 | | | | |
| INF01 | 0480 | 23 | 0 | | | | |
| INF02 | 0503 | 21 | 0 | | | | |
| PRINT | 0380 | 01 | 0 | | | | |
| RATE | 0302 | 03 | 0 | | | | |
| RATEX | 0402 | 10 | 0 | | | | |
| START | 0480 | 10 | 0 | | | | |
| TIMECD | 0300 | 80 | 0 | | | | |
| WORK1 | 0472 | 07 | 0 | | | | |
| 16 LABELS | | | | | | | |

Figure 1-3. An Overlay Program

INTRODUCTION

STATEMENT FORMAT

Character Set

The System Ten character set comprises the 64 characters in columns 2 thru 5 of the USASCII code chart (see figure 1-4).

The literal data field of a Define Memory statement may contain any of the 64 characters. Since this field is bounded by single apostrophes, an apostrophe within the field must be represented by two successive apostrophes.

* as an Address

An asterisk may be used in the operand field of instruction statements to signify the address of the instruction itself. It may be used in any of the following three forms:

```
*  
* - constant  
* + constant
```

For example, the Branch instruction

```
BC *+10(2),*+40(5)
```

states that if the Condition Code is 2, control passes to the instruction immediately following the Branch instruction; if the Condition Code is not 2, then control passes (unconditionally) to the fourth instruction following the Branch instruction.

An asterisk may also be used as an address constant in a Define Memory statement. Again, it may be used either by itself or with a constant (+ or -).

| COLUMN | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | |
|--------|---------|-----------------------|-----------------------|--------------------|---------|---|---|-------------|-------------|--------------|---------------|
| ROW | 7 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | | |
| | 6 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | | |
| | 5 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | | |
| | 4 3 2 1 | | | | | | | | | | |
| 0 | 0 0 0 0 | NUL 12-9-1 12-09-1 | DLE 11-9-1 12-11-1 | NO PUNCH 12-8-7 | SP 0 | 0 | 0 | @ 8-4 | P 11-7 | 8-1 | p 12-11-7 |
| 1 | 0 0 0 1 | SOH 12-9-2 | DC1 11-9-2 | ! | - | 1 | 1 | A 12-1 | Q 11-8 | a 12-0-1 | q 12-11-8 |
| 2 | 0 0 1 0 | STX 12-9-3 | DC2 11-9-3 | | 2 | 2 | 2 | B 12-2 | R 11-9 | b 12-0-2 | r 12-11-9 |
| 3 | 0 0 1 1 | ETX 12-9-4 | DC3 11-9-4 | # | 3 | 3 | 3 | C 12-3 | S 0-2 | c 12-0-3 | s 11-0-2 |
| 4 | 0 1 0 0 | EOT 9-7 | DC4 9-8-4 | \$ | 4 | 4 | 4 | D 12-4 | T 0-3 | d 12-0-4 | t 11-0-3 |
| 5 | 0 1 0 1 | ENQ 0-9-5 | NAK 9-8-5 | % | 5 | 5 | 5 | E 12-5 | U 0-4 | e 12-0-5 | u 11-0-4 |
| 6 | 0 1 1 0 | ACK 0-9-6 | SYN 9-2 | & | 6 | 6 | 6 | F 12-6 | V 0-5 | f 12-0-6 | v 11-0-5 |
| 7 | 0 1 1 1 | BEL 0-9-7 | ETB 0-9-6 | / | 7 | 7 | 7 | G 12-7 | W 0-6 | g 12-0-7 | w 11-0-6 |
| 8 | 1 0 0 0 | BS 11-9-6 | CAN 11-9-8 | (| 8 | 8 | 8 | H 12-8 | X 0-7 | h 12-0-8 | x 11-0-7 |
| 9 | 1 0 0 1 | HT 12-9-5 | EM 11-9-8-1 |) | 9 | 9 | 9 | I 12-9 | Y 0-8 | i 12-0-9 | y 11-0-8 |
| 10 | 1 0 1 0 | LF 0-9-5 | SUB 9-8-7 | * | : | : | : | J 11-1 | Z 0-9 | j 12-1-1 | z 11-0-9 |
| 11 | 1 0 1 1 | VT 12-9-3 | ESC 0-9-7 | + | ; | ; | ; | K 11-2 | [12-8-2 | k 12-11-2 | { 12-0 |
| 12 | 1 1 0 0 | FF 12-9-4 | FS 11-9-4 | , | < | < | < | L 12-8-4 | \ 0-8-2 | l 12-11-3 | 12-11 |
| 13 | 1 1 0 1 | CR 12-9-5 | GS 11-9-8-6 | - | = | = | = | M 11-4 |] 11-8-2 | m 12-11-4 | } 11-0 |
| 14 | 1 1 1 0 | SO 12-9-6 | RS 11-9-8-6 | . | > | > | > | N 11-5 | ^ 11-8-7 | n 12-11-5 | ~ 11-0-1 |
| 15 | 1 1 1 1 | SI 12-9-7 | US 11-9-8-7 | / | ? | ? | ? | O 11-6 | _ 0-8-5 | o 12-11-6 | DEL 12-9-7 |

△ Hollerith code

Figure 1-4. USASCII and Hollerith Codes

INTRODUCTION

SOURCE STATEMENT FIELDS

A source statement may contain up to 80 characters. It consists of some or all of the following fields in the order shown. The label, operation, operand, and comments fields are each separated from one another by at least one blank character.

| <u>FIELD</u> | <u>REMARKS</u> |
|------------------------|--|
| Label | Optional. First character begins in column 1. Exception: In debug statements, column 1 contains a \$-sign and the label field begins in column 2. |
| Operation | Required. Preceded and followed by at least one blank character. |
| Operand | Optional in some cases. Preceded by at least one and not more than eight blank characters. |
| Comments | Optional. Preceded by at least one blank. |
| Continuation | Must be blank. Column 70. |
| Program Identification | Optional. Columns 71-76. |
| Sequence Number | Optional. Columns 77-80. |

Label Field

This field, when permitted, is optional. Except for the assembler commands COMMON, NORMAL, END, EJECT, SPACE, and TITLE, any statement may be labeled.

If present, the label field consists of 1 to 6 characters, and must begin in column 1 of the statement. The first character must be from columns 4 or 5 of the USASCII code chart, while succeeding characters may be from columns 4, or 5 or 0-9 of column 3. The label field must be followed by at least one blank.

If the label is omitted, then column 1 of the statement must be blank to show that it is omitted. The only exception to this rule is the case of the comment statement which begins with an * in column 1.

Typically, the first character of a label is alphabetic, and the remaining characters are alphanumeric (A - Z, 0 - 9).

Two statements cannot have the same label.

NOTE

The following characters should not be used because they are reserved for software labels:

{\)^_

Also, the @ sign must not be used in the label field.

Operation Field

Required. This field consists of a keyword mnemonic. For example, the mnemonic for the Add instruction is the letter A, the mnemonic for the Form Numeric instruction is the pair of letters FN, etc. The mnemonic for each machine instruction is detailed in Section 2, Machine Instruction Statements, while those for the assembler commands are detailed in Section 3, Assembler Commands. The operation field must be immediately preceded and followed by at least one blank. If an operand follows, not more than 8 blanks may intervene.

The mnemonics for the machine instruction statements are:

| | |
|----|--------------------|
| A | Add |
| BC | Branch Conditional |
| C | Compare |
| D | Divide |
| E | Edit |
| X | Exchange |
| FN | Form Numeric |
| MC | Move Character |
| MN | Move Numeric |
| M | Multiply |
| R | Read |
| S | Subtract |
| W | Write |

The mnemonics for the assembler commands are:

COMMON
EJECT
END
EXEC
NORMAL
ORG
SPACE
TITLE
DM

INTRODUCTION

Operand Field

This field is not expected in the assembler commands COMMON, NORMAL, EJECT, and END. In the assembler commands ORG and SPACE it is optional. In all other assembler commands, and in all machine instructions, the operand field is required.

The operand field must be immediately preceded by 1-8 blanks. It is followed by at least one blank. Its format depends upon which mnemonic appears in the operation field. For example, an Add instruction requires an operand field of the "two length form". The various forms for the machine instructions and assembler commands are detailed in Section 2, Machine Instruction Statements, and Section 3, Assembler Commands.

Comments Field

Optional. Any statement may contain a comments field.

If present, it must be separated from the operand field by at least one blank, and must not extend beyond column 69. If a comments field appears in an ORG or SPACE assembler command which has no operand field, then it must be separated from the operation field by at least eight blanks. A comments field may contain any System Ten character (columns 2, 3, 4, or 5 of the USASCII code chart).

Program Identification Field

Optional. This consists of columns 71-76. Whatever is in this field of the first line is punched in columns 71-76 of every object card. This field has no effect upon, and does not appear in, the program listing.

Sequence Number Field

Optional. This consists of columns 77-80. Whatever is in this field is copied to the program listing under the heading SEQ.

STANDARD CODING FORMS AND SOURCE CARDS

Although the programmer is free to vary the spacing between fields, he may prefer to use standard System Ten assembler coding forms and source cards (figures 1-5 and 1-6). With readability in mind, they were designed with the label field beginning in column 1, the operation field in column 8, and the operand field in column 15. Column 70, which must always be blank, is shaded.

COMMENTS STATEMENT

A statement which begins with * in column 1 is treated as a comment. It is copied to the listing but has no other effect upon assembly.

DEBUG STATEMENT

A statement containing \$ in column 1 is called a Debug statement. A Debug statement is ignored by the assembler if the assembler parameter DEBUG=NO is in effect. If the assembler parameter DEBUG=YES is in effect, the \$ is ignored and the remainder of the statement is assembled. In such a case the label field begins in column 2 (instead of column 1, as in a non-Debug statement). If the label field is omitted, column 2 must be blank to show that it is omitted.

A card containing \$* in columns 1,2 is listed as a comment if the assembler parameter DEBUG=YES is in effect. The card is ignored if DEBUG=NO is in effect. DEBUG=NO is the usual default case, but it can be DEBUG=YES. The first card of the assembler deck contains parameters which determine the default.

The assembler parameters are discussed in Section 5 and Appendix A.

INSTRUCTION ALIGNMENT

Each machine instruction statement generates one instruction consisting of 10 characters. The assembler aligns each instruction so that when it is loaded, the leftmost character will have an address with a low order value of zero (as required by the hardware). This is sometimes referred to as tens boundary alignment. Adjacent instructions are aligned at adjacent tens boundaries. If the first of a series of instructions is preceded by a data field which does not terminate at a tens boundary, the instruction is aligned at the next higher tens boundary beyond the data field. If a machine instruction statement contains a label, a length of 10 is entered for that label in the label table. Index specification zero (indicating no indexing) is also entered.

OPERATION FIELD KEYWORDS

As with all assembler statements, the programmer writes a keyword in the operation field. There are thirteen keywords for machine instruction statements (corresponding to the thirteen machine instructions). For example, to assemble an Add instruction, the programmer writes "A" in the operation field; to assemble a Divide instruction, he writes "D", etc.

The operation field keyword determines the Op Code in the assembled instruction. It also determines the general operand form which the assembler will expect the programmer to use in writing his assembler statement. For example, the Add instruction always requires that the programmer use the "two length" form, while the Read instruction requires that the programmer use the "device/channel" form.

Five instructions require the "two length" form, five require the "one length" form, two require the "device/channel" form, and one requires the "branch variant" form. The assembler examines the operation field keyword before deciding which form to expect. Once the form is selected, the assembler proceeds to assemble the information supplied by the programmer, and to supply missing information where permitted. For each form, the assembler knows what the programmer must supply and what he may omit.

SYNTACTICAL ERRORS

An error listing is generated if there are deviations from the rules. In such a case the assembly is terminated after the label table has been listed.

INTRODUCTION

OPERAND FORMS

The four operand forms used in machine instruction statements are shown below.

| INSTRUCTION NAME | OP CODE | OPERAND FORM | FORM NAME |
|------------------|---------|-------------------|---------------------|
| Add | A | | |
| Divide | D | | |
| Form Numeric | FN | A(LA,IA),B(LB,IB) | Two-length form |
| Multiply | M | | |
| Subtract | S | | |
| Compare | C | | |
| Exchange | X | | |
| Edit | E | A(L,IA),B(,IB) | One-length form |
| Move Character | MC | | |
| Move Numeric | MN | | |
| Read | R | A(D,IA),B(C,IB) | Device/Channel form |
| Write | W | | |

BRANCH

| | | | |
|-----------------|----|-----------|-------------|
| Conditional | BC | A(V),B(V) | |
| Link | BC | A(6),B(V) | |
| Service Request | BC | A(7),B(0) | Branch form |
| Unconditional | BC | A(5) | |
| Switch | BC | A(8) | |

Legend: A,B Address-A, Address/Count-B (range 0-9999)

 C Channel and Control (range 0-5)

 D Input/Output Device (range 0-9)

 IA,IB Index Register (range 0-3, 0 = no indexing)

 L Length (range 1-100)

 LA,LB Length (range 1-10)

 V Branch Variant (range 0-9)

Operand Fields vs. Instruction Fields

A, LA, IA, B, LB, IB in the above forms are the same symbols used in the Programmers Machine Reference Manual for identifying the fields of the internal instruction format:

| | | | | | | | | | | |
|-----------|---|---|----|----|---|----|----|---|---|-----|
| CHARACTER | | | | | | | | | | BIT |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
| | F | | AC | IA | | IB | BC | | | 7 |
| | | | | | | | | | | 5 |
| | | | | | | | | | | 4 |
| LA | | A | | LB | | B | | | | 3 |
| | | | | | | | | | | 2 |
| | | | | | | | | | | 1 |

Whatever value the programmer indicates for A, LA, IA, B, LB, IB in his assembler statement will be assembled into the corresponding field of the internal instruction. D (device) is assembled into the LA field. C (channel) is assembled into the LB field. V (variant) is assembled into LA if its left bordering parenthesis adjoins A; it is assembled into LB if its left bordering parenthesis adjoins B.

Positional Aspects

The operand field contains only positional parameters and punctuation marks (parenthesis, comma). The symbols used in this part to present the general operand forms are not keyword parameters. They are used merely to remind the programmer of what information is expected and how this information is assembled into the internal machine instruction:

```
S A(LA,IA),B(LB,IB)
```

The keyword "S" calls for the Subtract Op Code 0111 and leads the assembler to expect the operand form shown. The form reminds the programmer to supply A and B (the addresses of Operand-A and Operand-B), LA and LB (the individual lengths of these operands), and the index register specifications, IA and IB (used for address modification). A typical statement might be:

```
S 400(5,1),500(10,0)
```


INTRODUCTION

In this example, 400 is assembled into the internal A field, 5 is assembled into the internal LA field, 1 is assembled into the internal IA field, etc.

Internal Blanks

The comma separating the left and right portions of the operand forms may be followed by trailing blanks (see figure 1-7).

LA, LB10

For instructions of the two length form, the programmer writes "10" when he wishes to specify a field length of 10 (he does not write 0). The assembler substitutes zero for 10 as required by the hardware in specifying a field length of 10. Thus, in the above example, the 10 causes the assembler to insert "0" into the internal LB instruction field.

L100

For the "one length" instructions where a field length of 100 is desired, the programmer writes "100" in the L position of the operand form. This causes the assembler to set zero into both the LA and LB fields of the internal machine instruction, thus conforming to the hardware convention for indicating a field length of 100.

Operand Constants and Symbols

The operand parameters inside the parentheses must all be constants. The address parameters, A and/or B, outside the parentheses must be one of the following:

```
*
*  constant

(flagged) constant
(flagged) constant  constant

symbol
symbol  constant
```

The use of operand constants and symbols is shown in figure 1-8. The rules governing a flaggable constant address are:

If the constant is not flagged and a COMMON statement is in effect, the common flag will be assembled into the internal machine instruction.

| PAGE 0001 10/12/71 (4-02) PROGRAM LISTING | | | | | | | | | | | | |
|---|------------|------------|------|-----|------|------|------|--------|------|----------------|-------------------|--|
| SEQ. | LOCN | INSTR/DATA | OP | A/R | L | I | B/S | L | I | LINE | LABEL | OPCODE OPERAND(S) AND/OR COMMENTS |
| 0000 | | | | | | | | | | 0001 | | * THIS EXAMPLE CALCULATES A BASEBALL PITCHER'S EARNED RUN AVERAGE. |
| 0000 | | | | | | | | | | 0002 | | * THE INPUT IS FROM CARDS WITH UP TO TEN SETS OF DATA ON A CARD. |
| 0000 | | | | | | | | | | 0003 | | * THE OUTPUT IS TO THE WORKSTATION. |
| 0000 | | | | | | | | | | 0004 | ORG 11 | |
| 0011 | | | | | | | | | | 0005 | XRI DM N4 | |
| 0015 | | | | | | | | | | 0007 | ORG 5000 | |
| 5000 | QU2J040011 | 13 | 5250 | 1 | 0 | 0011 | 4 | 0 | 0008 | INIT FN | ZERO, XRI | * ZEROS TO REGISTER |
| 5010 | 1528710080 | 00 | 5287 | 1 | 0 | 0080 | 1 | 0 | 0009 | R | INPUT(1), 80(1) | * READ A CARD FROM THE READER |
| 5020 | S5RU800000 | 11 | 5258 | 3 | 0 | 0000 | 0 | 0 | 0010 | BC | BLANKS(3) | * END JOB IF UNIT SEPERATOR. |
| 5030 | PUR8730258 | 14 | 5287 | 0 | 1 | 5258 | 3 | 0 | 0011 | LOOP C | IP, BLANKS | * IS REST OF CARD BLANK? |
| 5040 | R5PP000000 | 11 | 5000 | 2 | 0 | 0000 | 0 | 0 | 0012 | BC | INIT(2) | * YES, READ NEXT CARD. |
| 5050 | QU2JU35373 | 13 | 5250 | 1 | 0 | 5373 | 3 | 0 | 0013 | FN | ZERO, QUOT+6(3) | * CLEAR OUT LAST 3 POS. OF DIVIDEND |
| 5060 | SJ2X730376 | 13 | 5287 | 3 | 1 | 5376 | 3 | 0 | 0014 | FN | IP, DIVSOR(3) | * PUT IN LEFT MOST 3 POSITIONS. |
| 5070 | 1UR5135376 | 06 | 5251 | 1 | 0 | 5376 | 3 | 0 | 0015 | M | THREE, DIVSOR(3) | * RESULT IS FOUR POSITIONS. |
| 5080 | 1J29U4U376 | 04 | 5290 | 1 | 1 | 5376 | 4 | 0 | 0016 | A | FIP, DIVSOR | * ADD TO RESULT |
| 5090 | SJ2Y14U367 | 13 | 5291 | 3 | 1 | 5367 | 4 | 0 | 0017 | FN | RUNS, QUOT(4) | * LEADING ZERO ASSURED. |
| 5100 | 2UP5235368 | 06 | 5252 | 2 | 0 | 5368 | 3 | 0 | 0018 | M | TWOSEV, QUOT+1(3) | * DIVIDEND IS 0, 5 POSITIONS, AND |
| 5110 | 4U3+655367 | 05 | 5376 | 4 | 0 | 5367 | 5 | 0 | 0019 | D | DIVSOR, QUOT | * 3 TRAILING ZEROS FOR 3 DECIMAL PLACES |
| 5120 | T5R+000000 | 11 | 5220 | 4 | 0 | 0000 | 0 | 0 | 0020 | BC | ERRWRT(4) | * CHECK FOR OVERFLOW. |
| 5130 | 1U25655367 | 04 | 5256 | 1 | 0 | 5367 | 5 | 0 | 0021 | A | FIVE, QUOT | * ROUND OFF RESULT |
| 5140 | P526165380 | 08 | 5261 | 0 | 0 | 5380 | 6 | 0 | 0022 | MC | EDITDM, ERA | * MOVE IN EDIT MASK |
| 5150 | PU367+5380 | 12 | 5367 | 0 | 0 | 5380 | 4 | 0 | 0023 | E | QUOT(4), ERA | * PUT ANSWER IN WITH EDITING |
| 5160 | P538065273 | 08 | 5380 | 0 | 0 | 5273 | 6 | 0 | 0024 | MC | ERA, MSG+6 | * ANSWER TO MESSAGE |
| 5170 | 052U730001 | 01 | 5257 | 0 | 0 | 0001 | 3 | 0 | 0025 | W | CR(0), 1(3) | * CARRIAGE RETURN ON WORKSTATION. |
| 5180 | 052V710012 | 01 | 5267 | 0 | 0 | 0012 | 1 | 0 | 0026 | A | MSG(0), 12(1) | * WRITE ANSWER ON WORKSTATION. |
| 5190 | 1U25440011 | 04 | 5254 | 1 | 0 | 0011 | 4 | 0 | 0027 | A | EIGHT, XRI | * TO GET NEXT SET OF DATA |
| 5200 | PUR5420013 | 14 | 5254 | 0 | 0 | 0013 | 2 | 0 | 0028 | C | EIGHT(2), XRI+2 | * HAS THERE BEEN 10 SETS OF DATA? |
| 5210 | R5PP055030 | 11 | 5000 | 2 | 0 | 5030 | 5 | 0 | 0029 | BC | INIT(2), LOOP(5) | * YES, READ CARD; NO, PROCESS DATA. |
| 5220 | 052U730001 | 01 | 5257 | 0 | 0 | 0001 | 3 | 0 | 0030 | ERRWRT W | CR(0), 1(3) | * CARRIAGE RETURN ON WORKSTATION. |
| 5230 | 052W910008 | 01 | 5279 | 0 | 0 | 0008 | 1 | 0 | 0031 | W | ERRMSG(0), 8(1) | * WRITE OVERFLOW MESSAGE ON WORKSTATION |
| 5240 | U5RU800000 | 11 | 5258 | 5 | 0 | 0000 | 0 | 0 | 0032 | BC | BLANKS(5) | * END JOB |
| 5250 | 0 | | 0001 | | 0001 | | 0034 | ZERO | DM | N'0' | | |
| 5251 | 3 | | 0001 | | 0001 | | 0035 | THREE | DM | N'3' | | |
| 5252 | 27 | | 0001 | | 0002 | | 0036 | TWOSEV | DM | N'27' | | |
| 5254 | 80 | | 0001 | | 0002 | | 0037 | EIGHT | DM | N'80'(1) | | |
| 5256 | 5 | | 0001 | | 0001 | | 0038 | FIVE | DM | N'5' | | |
| 5257 | M | | 0001 | | 0001 | | 0039 | CR | DM | C'H' | | |
| 5258 | | | 0001 | | 0003 | | 0040 | BLANKS | DM | C3' | | |
| 5261 | 0+00- | | 0001 | | 0006 | | 0041 | EDITDM | DM | C' 0+00=1 | | |
| 5267 | ERA = 00.0 | | 0001 | | 0012 | | 0042 | MSG | DM | C'ERA = 00.00' | | |
| 5279 | OVERFLOW | | 0001 | | 0008 | | 0043 | ERRMSG | DM | C'OVERFLOW' | | |
| 5287 | | | 0000 | | 0080 | | 0044 | INPUT | DM | OC80 | | |
| 5287 | | | 0001 | | 0003 | | 0045 | IP | DM | N3(,1) | | * INNINGS PITCHED |
| 5290 | | | 0001 | | 0001 | | 0046 | FIP | DM | N1(,1) | | * FRACTIONAL INNINGS PITCHED=0,1,OR 2. |
| 5291 | | | 0001 | | 0003 | | 0047 | RUNS | DM | N3(,1) | | * EARNED RUNS ALLOWED |
| 5294 | | | 0001 | | 0073 | | 0048 | DM | DM | C73 | | * REMAINDER OF CARD IMAGE. |
| 5367 | | | 0001 | | 0005 | | 0049 | QUOT | DM | N5 | | * ANSWER OF DIVIDE WILL BE 5 POSITIONS. |
| 5372 | | | 0001 | | 0004 | | 0050 | DM | DM | N4 | | * DIVIDEND WILL INCLUDE THESE POSITIONS |
| 5376 | | | 0001 | | 0004 | | 0051 | DIVSOR | DM | N4 | | |
| 5380 | | | 0001 | | 0006 | | 0052 | ERA | DM | C6 | | * EARNED RUN AVERAGE |
| 5386 | | | 5000 | | | | 0054 | EXEC | INIT | | | |
| 5386 | | | | | | | 0055 | END | | | | |

Figure 1-7. Operands with Internal Blanks

INTRODUCTION

If a "C" is appended to the constant, the corresponding common flag will be assembled into the internal machine instruction even though a NORMAL statement is in effect.

Conversely, if a "P" is appended to the constant, the corresponding common flag will not be assembled into the internal machine instruction even though a COMMON statement may be in effect.

Implied Operand Forms

The programmer is free to omit length parameters (LA, LB, L) and index parameters (IA, IB) from machine instruction statements, leaving it to the assembler to supply these values from the label table if the corresponding operand address is symbolic. If the operand address is a constant or *, the value supplied by the assembler for an omitted length parameter is 1, and the value supplied for an omitted index specification is 0. If L is omitted, the value supplied by the assembler is determined by A (and in no way depends upon B). Thus, if A is a constant and L is omitted, the assembler acts as though the programmer has written "1" in the L position. If A is a symbol, the assembler supplies the length from the label table entry corresponding to the symbol.

It is not permissible to omit D (Device) or C (Channel) from the Read or Write instruction statement, nor is it permissible to omit V (Variant) from the Branch instruction.

When a length parameter is omitted, and the index parameter is written, the comma which would have separated the two parameters is written adjacent to the left parenthesis and the index specification follows immediately with no intervening blanks:

```
S ALFA(,1),BETA(,2)
```

When a length parameter is written, and the index parameter is omitted, the comma which would have separated the two parameters is also omitted. The closing right parenthesis follows immediately after the length specifications:

```
S ALFA(5),BETA(7)
```

When both the length parameter and the index parameter are omitted, the parenthesis and the comma internal to the parenthesis are also omitted:

```
S ALFA,BETA
```

| PAGE 0001 10/12/71 (4=02) PROGRAM LISTING FOR EXAMPLES OF OPERAND CONSTANTS AND SYMBOLS | | | | | | | | | | | | | |
|---|------------|------------|-------|-----|------|---|-------|------|---|------|-------|--------|---|
| SEQ. | LOCN | INSTR/DATA | OP | A/R | L | I | B/S | L | I | LINE | LABEL | UPCODE | OPERAND(S) AND/OR COMMENTS |
| 0000 | | | | | | | | | | 0002 | | URG | 300 |
| 0300 | PPS0010350 | 14 | 0300 | | 0 | 0 | 0350 | 1 | 0 | 0003 | START | C | *,BRSW DOES FIRST POSITION OF BRSW='P' |
| 0310 | R0ST050320 | 11 | 0340 | | 2 | 0 | 0320 | 5 | 0 | 0004 | | BC | RESET(2),**+10(5) YES, NO |
| 0320 | P032010350 | 08 | 0320 | | 0 | 0 | 0350 | 1 | 0 | 0005 | | MC | *,BRSW PUT 'P' INTO FIRST POSITION |
| 0330 | U0SU000000 | 11 | 0350 | | 5 | 0 | 0000 | 0 | 0 | 0006 | | BC | BRSW(5) |
| 0340 | P04U810350 | 09 | 0458 | | 0 | 0 | 0350 | 1 | 0 | 0007 | RESET | MN | FIVE,BRSW MAKE BRANCH UNCONDITIONAL |
| 0350 | U0SV050400 | 11 | 0360 | | 5 | 0 | 0400 | 5 | 0 | 0008 | BRSW | BC | **+10(5),**+50(5) FALLS THRU EVERY OTHER TIME |
| 0360 | PPS6510459 | 14 | 0365 | | 0 | 0 | 0459 | 1 | 0 | 0009 | | C | **+5,SW IS SW ONE? |
| 0370 | R0SY050380 | 11 | 0390 | | 2 | 0 | 0380 | 5 | 0 | 0010 | | BC | END(2),**+10(5) YES, NO |
| 0380 | P038510459 | 08 | 0385 | | 0 | 0 | 0459 | 1 | 0 | 0011 | | MC | **+5,SW PUT A ONE INTO SW |
| 0390 | 1P39030455 | 04 | 0390 | | 1 | 0 | 0455 | 3 | 0 | 0012 | END | A | *,CTR ADD ONE TO CTR |
| 0400 | 1PTP030455 | 07 | 0400 | | 1 | 0 | 0455 | 3 | 0 | 0013 | | S | *,CTR SUBTRACT ONE FROM CTR |
| 0410 | 2P45330455 | 04 | 0453 | | 2 | 0 | 0455 | 3 | 0 | 0014 | | A | NUM+2(2),CTR ADD '34' TO CTR |
| 0420 | P04553100P | 08 | 0455 | | 0 | 0 | 1000C | 3 | 0 | 0015 | | MC | CTR, 1000C MOVE CTR TO POS. 1000 IN COMMON |
| 0430 | P200P14000 | 08 | 2000C | | 0 | 0 | 4000 | 1 | 0 | 0016 | | MC | 2000C,4000P COMMON ADR. TO PARTITION ADR. |
| 0440 | U0TU000000 | 11 | 0450 | | 5 | 0 | 0000 | 0 | 0 | 0017 | | BC | STOP(5) |
| 0450 | | | 0001 | | 0001 | | | 0018 | | 0018 | STOP | DM | C' ' TO STOP RUN |
| 0451 | 1234 | | 0001 | | 0004 | | | 0019 | | 0019 | NUM | DM | N4'1234' |
| 0455 | 0 | | 0001 | | 0003 | | | 0020 | | 0020 | CTR | DM | N3'0' |
| 0458 | 5 | | 0001 | | 0001 | | | 0021 | | 0021 | FIVE | DM | N'5' |
| 0459 | 0 | | 0001 | | 0001 | | | 0022 | | 0022 | SW | DM | N'0' |
| 0460 | | | 0300 | | | | | 0023 | | 0023 | EXEC | START | |
| 0460 | | | | | | | | 0024 | | 0024 | END | | |

| PAGE 0001 10/12/71 (4=02) LABEL TABLE FOR EXAMPLES OF OPERAND CONSTANTS AND SYMBOLS | | | | |
|---|------|------|------|----------|
| LABEL | LOC | LNTH | INDX | MESSAGES |
| BRSW | 0350 | 10 | 0 | |
| CTR | 0455 | 03 | 0 | |
| END | 0390 | 10 | 0 | |
| FIVE | 0458 | 01 | 0 | |
| NUM | 0451 | 04 | 0 | |
| RESET | 0340 | 10 | 0 | |
| START | 0300 | 10 | 0 | |
| STOP | 0450 | 01 | 0 | |
| SW | 0459 | 01 | 0 | |
| 9 LABELS | | | | |

Figure 1-8. Operand Constants and Symbols

INTRODUCTION

INSTRUCTION FIELDS AND HOW THEY ARE GENERATED

The preceding paragraphs, while they have dealt with the internal instruction fields, have been mainly concerned with the machine instruction assembler statements. The following paragraphs will be concerned with the instruction fields as such and how they are generated by the assembler.

Each ten-character instruction comprises nine fields in the format shown below:

F, A, LA, IA, AC, B, LB, IB, BC

For each machine instruction statement, the assembler generates one complete instruction.

| CHARACTER | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | BIT |
|-----------|----|---|---|---|----|----|----|---|----|----|-----|
| | | F | | | AC | | IA | | IB | BC | 7 |
| | | | | | | | | | | | 5 |
| | | | | | | | | | | | 4 |
| | | | | | | | | | | | 3 |
| | LA | | A | | | LB | | | B | | 2 |
| | | | | | | | | | | | 1 |

F (Op Code)

The operation field keyword determines the Op Code. Invariably, "R", the keyword for Read, is assembled as 0000, the Op Code for Read; "W", the keyword for Write, is assembled as 0001, the Op Code for Write; etc.

| OP CODE | F | INSTRUCTION NAME |
|---------|------|------------------|
| R | 0000 | Read |
| W | 0001 | Write |
| | 0010 | (Not Used) |
| | 0011 | (Not Used) |
| A | 0100 | Add |
| D | 0101 | Divide |
| M | 0110 | Multiply |
| S | 0111 | Subtract |
| MC | 1000 | Move Character |
| MN | 1001 | Move Numeric |
| | 1010 | (Not Used) |
| BC | 1011 | Branch |
| E | 1100 | Edit |
| FN | 1101 | Form Numeric |
| C | 1110 | Compare |
| X | 1111 | Exchange |

A, B (Operand - Address/Count)

Instruction fields A and B are derived from assembler operand parameters A and B, respectively.

If the operand parameter is a symbol, the address associated with the symbol in the label table is inserted into the instruction field. If the operand parameter is a symbol plus/minus a constant, the constant is used to adjust the value from the label table; the sum/difference is inserted into the instruction field.

If the operand parameter is *, the address of the instruction being assembled is inserted into the instruction field. If the operand parameter is * plus/minus a constant, the constant is used to adjust the address of the current instruction; the sum/difference is inserted into the instruction field.

If the operand parameter is a constant, the constant is inserted into the instruction field. If the operand parameter is a constant plus/minus a constant, the sum/difference is inserted into the instruction field.

AC, BC (Common/Partition)

Instruction fields AC and BC are determined by assembler parameters A and B, respectively.

If the assembler parameter A is symbolic,

e.g. ALPHA
ALPHA+100

the AC field is set ON if the address represented by the symbol is in COMMON. Otherwise, AC is set OFF.

If the assembler parameter A is *,

e.g. *
*+100

the AC field is set ON if a COMMON statement is in effect. Otherwise, AC is set OFF.

If the assembler parameter A is an unflagged constant,

e.g. 500
500+10

the AC instruction field is set ON if a COMMON statement is in effect. Otherwise, AC is set OFF.

INTRODUCTION

If the assembler parameter A is a constant flagged with a C,

e.g. 500C
500C+10

the AC instruction field is set ON, unconditionally.

If the assembler parameter A is a constant flagged with P,

e.g. 500P
500P+10

the AC instruction field is set OFF, unconditionally.

The above rules showing how AC is derived from A apply equally in showing the relation between B and BC. It is merely necessary for the reader to substitute "B" when he reads "A" and to substitute "BC" when he reads "AC".

The COMMON statement is described in the section entitled Assembler Commands.

IA, IB (Index Register Specification)

Instruction fields IA and IB are taken from assembler statement parameters IA and IB, providing these parameters are not omitted. If IA is omitted from the assembler statement, then the value of IA is supplied by the assembler. The value the assembler supplies depends upon assembler statement parameter A. A parallel relation holds between IB and B when IB is omitted from the assembler statement. Because of this parallelism, the rules will be presented only for IA.

If IA is omitted from an assembler statement, and if A is a constant or *, then instruction field IA is set to 0 (indicating no indexing).

If IA is omitted from an assembler statement, and if A is a symbol for the address of an instruction, then instruction field IA is set to 0 (indicating no indexing).

If IA is omitted from an assembler statement, and if A is a symbol for the address of a data field, then instruction field IA is taken from the label table index entry for that symbol. The establishment of a data field index entry in the label table is part of the function of the Define Memory Statement (described in the section entitled Assembler Commands).

LA, LB (Length, Device, Channel, Variant)

For the "device/channel" form, instruction field LA is taken from D, and instruction field LB is taken from C.

For the "branch" form, instruction field LA is taken from the value of V in the parenthesis adjacent to assembler statement parameter A. LB is taken from the value of V in the parenthesis adjacent to assembler statement parameter B.

For the two length form, instruction field LA is taken from the value of LA in the assembler statement. If assembler parameter LA=10, it is assembled as O. In similar fashion, instruction field LB is taken from assembler statement parameter LB. If LA or LB are omitted, the assembler supplies the values for the omitted parameters as described under "Implied Forms", above.

For the one length form, both instruction fields LA and LB are taken from assembler parameter L. LA is set to the tens digit of L, and LB is set to the units digit of L. If L=100, both LA and LB are set to O. If L is omitted, the assembler supplies a value for it as described under "Implied Forms", above.

ASSEMBLER COMMANDS

Unlike the machine instruction statements, the assembler commands do not generate executable code. The primary purpose is to control the assembly. An exception to this is the DM (Define Memory) statement which also permits the programmer to specify data to be loaded with the program.

Listing Control Commands

TITLE establishes the title line text of the assembly listing. This line is printed at the top of each page.

SPACE generates blank lines in the assembly listing, thus enabling the programmer to make his listing more easily readable.

EJECT advances the assembly listing to the next page.

INTRODUCTION

Address Control Commands

- ORG sets the value for the next address to be assigned by the assembler.
- NORMAL causes the assembler to start assigning addresses in partition.
- COMMON causes the assembler to start assigning addresses in common.

Termination Control Commands

- EXEC generates an S (start) card in the object code. The S card will be interpreted during loading. It causes the loader to stop loading and to begin execution at the address specified by the programmer. For source programs containing several overlays, the final statement in each overlay is an EXEC statement.
- END shows the end of the source program and is thus the last statement in it. Each source program, even if it contains several overlays, may contain but one END statement.

Memory Allocation Commands

- DM defines memory data fields and generates initial data.

Section 2

MACHINE INSTRUCTION STATEMENTS

INTRODUCTION
ADD
BRANCH (Conditional, Unconditional, Switch)
BRANCH AND LINK
BRANCH ON SERVICE REQUEST
COMPARE
DIVIDE
EDIT
EXCHANGE
FORM NUMERIC
MOVE CHARACTER
MOVE NUMERIC
MULTIPLY
READ
SUBTRACT
WRITE

MACHINE INSTRUCTION STATEMENTS

INTRODUCTION

This section details the information related to the machine instruction statements, including information about their execution. After the introduction, the page headings are arranged in alphabetical order for ease of reference:

- ADD
- BRANCH (Conditional, Unconditional, Switch)
- BRANCH AND LINK
- BRANCH ON SERVICE REQUEST
- COMPARE
- DIVIDE
- EDIT
- EXCHANGE
- FORM NUMERIC
- MOVE CHARACTER
- MOVE NUMERIC
- MULTIPLY
- READ
- SUBTRACT
- WRITE

The range shown in the assembler statements (0-9999) is usually represented symbolically when defined as a label in an assembler statement.

ASSEMBLER STATEMENT

Operation Mnemonic

A

Operand Form

A(LA,IA),B(LB,IB) Two-length form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|--------|----------|------------------------------|
| A | 0-9999 | REQUIRED | Address of Operand-A |
| LA | 1-10 | Optional | Length of Operand-A |
| IA | 0-3 | Optional | Index Register for A |
| B | 0-9999 | REQUIRED | Address of Operand-B and Sum |
| LB | 1-10 | Optional | Length of Operand-B and Sum |
| IB | 0-3 | Optional | Index Register for B |

Notes

For a discussion of the values supplied by the assembler when the programmer omits an optional parameter from the operand specification, see Section 1, under "Implied Operand Forms."

EXECUTION

Effect

The Add instruction (Op Code 0100) adds the numeric portions of Operand-A and Operand-B algebraically. The sum replaces Operand-B.

Details

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.

The instruction A *,FIELD will add 1 to the contents of the label FIELD.

Condition Codes

After completion of the Add instruction:

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

BRANCH INSTRUCTION

The Branch instruction (Op Code 1011) 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.

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.

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.

ASSEMBLER STATEMENT

Operation Mnemonic

BC

Operand Form

A(V),B(V) Branch-Conditional form
 A(5) Unconditional-branch form
 A(8) Unconditional-switch-and-branch form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|---------|----------|-----------------------|
| A | 0-9999 | REQUIRED | First branch address |
| V | 0-5,8,9 | REQUIRED | Variant |
| B | 0-9999 | OPTIONAL | Second branch address |

NOTE: If B is used, it must be in the form B(V).

EXECUTION

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 V (and thus 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 "Branch and 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").

Condition Code

The Branch instruction does not alter the Condition Code.

ASSEMBLER STATEMENT

Operation Mnemonic

BC

Operand Form

A(6),B(V) Branch-and-link form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|---------|----------|----------------|
| A | 0-9999 | REQUIRED | Return address |
| V | 0-5,8,9 | REQUIRED | Variant |
| B | 0-9999 | REQUIRED | Start address |

Notes

"6" is assembled into the LA instruction field.
V may assume the following values:

- 0---No branch and no link.
- 1---Branch and link if Condition Code is 1.
- 2---Branch and link if Condition Code is 2.
- 3---Branch and link if Condition Code is 3.
- 4---Branch and link if Condition Code is 4.
- 5---Branch and link, unconditionally.
- 8---Switch,branch, and link, unconditionally.
- 9---No branch and no link.

EXECUTION

Effect

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

If V (thus 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 V (thus 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).

Condition Code

Branch and Link does not alter the Condition Code.

ASSEMBLER STATEMENT

Operation Mnemonic

BC

Operand Form

A(7),B(0) Branch-on-service-request form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|--------|----------|--------------------------------|
| A | 0-9999 | REQUIRED | Save address for device number |
| B | 0-9999 | REQUIRED | Branch address |

EXECUTION

Effect

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."

Storing Device Number

"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 Code

Branch on Service Request does not alter the Condition Code.

ASSEMBLER STATEMENT

Operation Mnemonic

C

Operand Form

A(L,IA),B(,IB) One-length form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|--------|----------|-----------------------------------|
| A | 0-9999 | REQUIRED | Address of Operand-A |
| L | 1-100 | Optional | Length of Operand-A and Operand-B |
| IA | 0-3 | Optional | Index Register for A |
| B | 0-9999 | REQUIRED | Address of Operand-B |
| IB | 0-3 | Optional | Index Register for B |

Notes

Instruction field LA is set to the tens digit of L, and instruction field LB is set to the units digit of L. If L=100, both LA and LB are set to 0. If L is omitted, the assembler supplies a value for it. To this value, the above rule holds.

For a discussion of values supplied by the assembler when the programmer omits an optional parameter from the operand specification, see Section 1, under "Implied Operand Forms."

EXECUTION

Effect

The Compare instruction (Op Code 1110) compares two fields and sets the Condition Code to indicate the relation between them. (This comparison is based upon the bit configurations for the System Ten character set as shown in figure 2-1.)

Details

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 or the rightmost position has been compared.

When the characters differ, Condition Code 1, or 3 and 4 is set ON (indicating that Operand-A is smaller or 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.

| 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 -0 |
| 1 | 1 | 0 | 0 | 0 | 1 | Q -1 |
| 1 | 1 | 0 | 0 | 1 | 0 | R -2 |
| 1 | 1 | 0 | 0 | 1 | 1 | S -3 |
| 1 | 1 | 0 | 1 | 0 | 0 | T -4 |
| 1 | 1 | 0 | 1 | 0 | 1 | U -5 |
| 1 | 1 | 0 | 1 | 1 | 0 | V -6 |
| 1 | 1 | 0 | 1 | 1 | 1 | W -7 |
| 1 | 1 | 1 | 0 | 0 | 0 | X -8 |
| 1 | 1 | 1 | 0 | 0 | 1 | Y -9 |
| 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 |

Figure 2-1. System Ten Character Set

ASSEMBLER STATEMENT

Operation Mnemonic

D

Operand Form

A(LA,IA),B(LB,IB) Two-length form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|--------|----------|---|
| A | 0-9999 | REQUIRED | Address of Operand-A, the divisor |
| LA | 1-10 | Optional | Length of divisor |
| IA | 0-3 | Optional | Index Register for A |
| B | 0-9999 | REQUIRED | Address of Operand-B, the dividend and the quotient |
| LB | 1-10 | Optional | Length of quotient |
| IB | 0-3 | Optional | Index Register for B |

Notes

For a discussion of the values supplied by the assembler when the programmer omits an optional parameter from the operand specifications, see Section 1, under "Implied Operand Forms."

EXECUTION

Effect

The Divide instruction (op Code 0101) computes the algebraic quotient (and remainder) of two operands.

Details

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 of the quotient field 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.

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.

Condition Codes

After completion of the Divide instruction:

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

ASSEMBLER STATEMENT

Operation Mnemonic

E

Operand Form

A(L,IA),B(,IB) One-length form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|--------|----------|---|
| A | 0-9999 | REQUIRED | Address of Operand-A |
| L | 1-100 | Optional | Length of Operand-A |
| IA | 0-3 | Optional | Index Register for A |
| B | 0-9999 | REQUIRED | Address of Operand-B, the control field |
| IB | 0-3 | Optional | Index Register for B |

Notes

Instruction field LA is set to the tens digit of L, and instruction field LB is set to the units digit of L. If L=100, both LA and LB are set to 0. If L is omitted, the assembler supplies a value for it. To this value, the above rule holds.

For a discussion of values supplied by the assembler when the programmer omits an optional parameter from the operand specification, see Section 1, under "Implied Operand Forms."

EXECUTION

Effect

The Edit instruction (Op Code 1100) 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.

Details

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.

Control Field (Operand-B)

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 control field 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 control field 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.

Edit Instruction 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 | bbbbbb123.45 | after editing |

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

ASSEMBLER STATEMENT

Operation Mnemonic

X

Operand Form

A(L,IA),B(,IB) One-length form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|--------|----------|-----------------------------------|
| A | 0-9999 | REQUIRED | Address of Operand-A |
| L | 1-100 | Optional | Length of Operand-A and Operand-B |
| IA | 0-3 | Optional | Index Register for A |
| B | 0-9999 | REQUIRED | Address of Operand-B |
| IB | 0-3 | Optional | Index Register for B |

Notes

Instruction field LA is set to the tens digit of L, and instruction field LB is set to the units digit of L. If L=100, both LA and LB are set to 0. If L is omitted, the assembler supplies a value for it. To this value, the above rule holds.

For a discussion of values supplied by the assembler when the programmer omits an optional parameter from the operand specification, see Section 1, under "Implied Operand Forms."

EXECUTION

Effect

The Exchange instruction (Op Code 1111) interchanges the characters in two fields of equal length. Each field can comprise 1-100 characters.

Details

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.

ASSEMBLER STATEMENT

Operation Mnemonic

FN

Operand Form

A(LA,IA),B(LB,IB) Two-length form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|--------|----------|----------------------|
| A | 0-9999 | REQUIRED | Address of Operand-A |
| LA | 1-10 | Optional | Length of Operand-A |
| IA | 0-3 | Optional | Index Register for A |
| B | 0-9999 | REQUIRED | Address of Operand-B |
| LB | 1-10 | Optional | Length of Operand-B |
| IB | 0-3 | Optional | Index Register for B |

Notes

For a discussion of values supplied by the assembler when the programmer omits an optional parameter from the operand specification, see Section 1, under "Implied Operand Forms."

EXECUTION

Effect

The Form Numeric instruction (Op Code 1101) 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.

Details

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 (0-9), 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.

ASSEMBLER STATEMENT

Operation Mnemonic

MC

Operand Form

A(L,IA),B(,IB) One-length form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|--------|----------|-----------------------------------|
| A | 0-9999 | REQUIRED | Address of Operand-A |
| L | 1-100 | Optional | Length of Operand-A and Operand-B |
| IA | 0-3 | Optional | Index Register for A |
| B | 0-9999 | REQUIRED | Address of Operand-B |
| IB | 0-3 | Optional | Index Register for B |

Notes

Instruction field LA is set to the tens digit of L, and instruction field LB is set to the units digit of L. If L=100, both LA and LB are set to 0. If L is omitted, the assembler supplies a value for it. To this value, the above rule holds.

For a discussion of values supplied by the assembler when the programmer omits an optional parameter from the operand specification, see Section 1, under "Implied Operand Forms."

EXECUTION

Effect

The Move Character instruction (Op Code 1000) moves a field containing 1-100 characters from one location in memory to another.

Details

Operand-A is copied into Operand-B, one character at a time, starting with the leftmost character, and proceeding from left to right.

Condition Code

2, after completion of Move Character instruction.

ASSEMBLER STATEMENT

Operation Mnemonic

MN

Operand Form

A(L,IA),B(,IB) One-length form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|--------|----------|-----------------------------------|
| A | 0-9999 | REQUIRED | Address of Operand-A |
| L | 1-100 | Optional | Length of Operand-A and Operand-B |
| IA | 0-3 | Optional | Index Register for A |
| B | 0-9999 | REQUIRED | Address of Operand-B |
| IB | 0-3 | Optional | Index Register for B |

Notes

Instruction field LA is set to the tens digit of L, and instruction field LB is set to the units digit of L. If L=100, both LA and LB are set to 0. If L is omitted, the assembler supplies a value for it. To this value, the above rule holds.

For a discussion of values supplied by the assembler when the programmer omits an optional parameter from the operand specification, see Section 1, under "Implied Operand Forms."

EXECUTION

Effect

The Move Numeric instruction (Op Code 1001) moves the numeric portion of a field containing 1-100 characters from one location in memory to another.

Details

The numeric portion of Operand-A is copied into Operand-B. The operation proceeds one digit at a time, starting with the leftmost digit and proceeding from left to right. The zone bits of Operand-B are undisturbed.

Condition Code

2, after completion of Move Numeric instruction.

ASSEMBLER STATEMENT

Operand Mnemonic

M

Operand Form

A(LA,IA),B(LB,IB) Two-length form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|--------|----------|--|
| A | 0-9999 | REQUIRED | Address of Operand-A, the multiplicand |
| LA | 1-10 | Optional | Length of multiplicand |
| IA | 0-3 | Optional | Index Register for A |
| B | 0-9999 | REQUIRED | Address of Operand-B, the multiplier and product |
| LB | 1-10 | Optional | Length of multiplier |
| IB | 0-3 | Optional | Index Register for B |

Notes

For a discussion of the values supplied by the assembler when the programmer omits an optional parameter from the operand specification, see Section 1, under "Implied Operand Forms."

EXECUTION

Effect

The Multiply instruction (Op Code 0110) computes the algebraic product of two 1 to 10 position numeric operands.

Details

Operand-A is the multiplicand.

Operand-B is the multiplier.

The product is developed in a field of length LA+LB starting at the Operand-B address. The product field may be thought of as consisting of two fields: the multiplier and a rightward extension with a length equal to that of the multiplicand.

If the factors differ in sign, bit-7 of the rightmost positions of the product is turned ON. If the signs are alike, bit-7 is turned OFF. Bit-5 is turned ON for all positions of the product. Bit-7 is always turned OFF for all positions of the product except the rightmost.

Condition Codes

After completion of the Multiply instruction:

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

ASSEMBLER STATEMENT

Operation Mnemonic

R

Operand Form

A(D,IA),B(C,IB) Device/Channel form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|--------|----------|--|
| A | 0-9999 | REQUIRED | Address of input area |
| D | 0-9 | REQUIRED | Device number |
| IA | 0-3 | Optional | Index Register for A |
| B | 0-9999 | REQUIRED | Count or indirect disc address (must be a numeric value) |
| C | 0-5 | REQUIRED | Channel and Control |
| IB | 0-3 | Optional | Index Register for B |

Notes

For a discussion of the values supplied by the assembler when the programmer omits an optional parameter from the operand specification, see Section 1 under "Implied Operand Forms."

Channel Specification

- 0. FAC, File Access Channel.
- 1. IOC, Input Output Channel, fill mode.
- 5. IOC, Input Output Channel, non-fill mode.

Device Specification

IOC devices are interchangeable in device number. Most installations prefer a standard setup. FAC device numbers are fixed.

STANDARD DEVICE NUMBERS

| IOC | FAC |
|---------------------|--------------------------------|
| 0 Workstation | Disc Controller |
| 1 Card Reader | Magnetic Tape Drive 1 |
| 2 Line Printer | Magnetic Tape Drive 2 |
| 3 Paper Tape Reader | Magnetic Tape Drive 3 |
| 4 Card Punch | Magnetic Tape Drive 4 |
| 5 - | - |
| 6 Paper Tape Punch | - |
| 7 - | - |
| 8 - | On Line Communications Adapter |
| 9 - | - |

Disc Address

B is the character count for any device except the disc controller. For the disc controller, the count is always 100. To select the disc controller, the programmer makes both D and C (Device and Channel) equal to zero. When the disc is thus specified, B becomes the indirect disc address, pointing to the 6 character disc address field. The format of the disc address field is:

| Character | 1 | 2 | 3 | 4 | 5 | 6 | Bit |
|-----------|---|---|---|---|---|---|-----|
| | A | B | D | E | F | G | 4 |
| | A | B | D | E | F | G | 3 |
| | A | B | D | E | F | G | 2 |
| | A | C | D | E | F | G | 1 |

In the above representation, the zone bits are not shown. Bit-5 must be ON. Bit-7 is ignored.

- A: Device number (0-9).
- B: Arm number (0-4).
- C: Hundreds digit (0 or 1) of a three-digit track number.
- D: Tens digit (0-9) of a three-digit track number.
- E: Units digit (0-9) of a three-digit track number.
- F: Tens digit (0-9) of a two-digit sector number.
- G: Units digit (0-9) of a two-digit sector number.

EXECUTION

Effect

The Read instruction (Op Code 0000) moves data from an input device to sequential locations in Main Memory.

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 that 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.

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), and 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 preempt 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 free if it is not bound to another partition. It is bound 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

For more information on Condition Codes and their meaning for particular devices, the reader is referred to Appendix F, "I/O Device Condition Codes."

ASSEMBLER STATEMENT

Operation Mnemonic

S

Operand Form

A(LA,IA),B(LB,IB) Two-length form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|--------|----------|--|
| A | 0-9999 | REQUIRED | Address of Operand-A (the subtrahend) |
| LA | 1-10 | Optional | Length of Operand-A |
| IA | 0-3 | Optional | Index Register for A |
| B | 0-9999 | REQUIRED | Address of Operand-B (the minuend) and the difference |
| LB | 1-10 | Optional | Length of Operand-B and difference |
| IB | 0-3 | Optional | Index Register for B |

Notes

For a discussion of the values supplied by the assembler when the programmer omits an optional parameter from the operand specification, see Section 1, under "Implied Operand Forms."

EXECUTION

Effect

The Subtract instruction (Op Code 0111) computes the algebraic difference between two operands. The difference replaces the minuend (Operand-B).

Details

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.

The instruction S *,FIELD will subtract 1 from the contents of the label FIELD.

Condition Codes

After completion of the Subtract instruction.

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

ASSEMBLER STATEMENT

Operation Mnemonic

W

Operand Form

A(D,IA),B(C,IB) Device/Channel form

Legend

| PARAMETER | RANGE | REQ/OPT | DESCRIPTION |
|-----------|--------|----------|--|
| A | 0-9999 | REQUIRED | Address of output area |
| D | 0-9 | REQUIRED | Device number |
| IA | 0-3 | Optional | Index Register for A |
| B | 0-9999 | REQUIRED | Count or indirect disc address (must be a numeric value) |
| C | 0-5 | REQUIRED | Channel and Control |
| IB | 0-3 | Optional | Index Register for B |

Notes

For a discussion of the values supplied by the assembler when the programmer omits an optional parameter from the operand specification, see Section 1 under "Implied Operand Forms."

Channel Specification

0. FAC, File Access Channel
1. IOC, Input Output Channel
2. FAC, Write Control
3. IOC, Write Control

Device Specification

IOC devices are interchangeable in device number. Most installations prefer a standard setup. FAC device numbers are fixed.

STANDARD DEVICE NUMBERS

| IOC | FAC |
|---------------------|--------------------------------|
| 0 Workstation | Disc Controller |
| 1 Card Reader | Magnetic Tape Drive 1 |
| 2 Line Printer | Magnetic Tape Drive 2 |
| 3 Paper Tape Reader | Magnetic Tape Drive 3 |
| 4 Card Punch | Magnetic Tape Drive 4 |
| 5 - | - |
| 6 Paper Tape Punch | - |
| 7 - | - |
| 8 - | On Line Communications Adapter |
| 9 - | - |

Disc Address

B is the character count for any device except the disc controller. For the disc controller, the count is always 100. To select the disc controller, the programmer makes both D and C (Device and Channel) equal to zero. When the disc is thus specified, B becomes the indirect disc address, pointing to the 6 character disc address field. The format of the disc address field is:

| Character | 1 | 2 | 3 | 4 | 5 | 6 | Bit |
|-----------|---|---|---|---|---|---|-----|
| | A | B | D | E | F | G | 4 |
| | A | B | D | E | F | G | 3 |
| | A | B | D | E | F | G | 2 |
| | A | C | D | E | F | G | 1 |

In the above representation, the zone bits are not shown. Bit-5 must be ON. Bit-7 is ignored.

A: Device number (0-9).

B: Arm number (0-4).

C: Hundreds digit (0 or 1) of a three-digit track number.

D: Tens digit (0-9) of a three-digit track number.

E: Units digit (0-9) of a three-digit track number.

F: Tens digit (0-9) of a two-digit sector number.

G: Units digit (0-9) of a two-digit sector number.

EXECUTION**Effect**

The Write instruction (Op Code 0001) 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.

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.

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. Service to the IOCs continues except during transmission of data between disc and memory.

Disc Access Sequence

A Write instruction addressing the disc does not typically preempt 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 free if it is not bound to another partition. It is bound 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

A Write instruction may specify the transmission of control characters to the external input/ output device by having bit-2 of the IB 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.

For more information on Write Control, the reader is referred to Appendix E, "I/O Device Control Characters."

Condition Codes

After completion of the Write instruction:

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

For more information on Condition Codes and their meaning for particular devices, the reader is referred to Appendix F, "I/O Device Condition Codes."

Section 3
ASSEMBLER COMMANDS

INTRODUCTION
COMMON
DM
EJECT
END
EXEC
NORMAL
ORG
SPACE
TITLE

ASSEMBLER COMMANDS

INTRODUCTION

This section details the information related to the assembler commands. After the introduction, the page headings are arranged in alphabetical order for ease of reference:

- COMMON
- DM
- EJECT
- END
- EXEC
- NORMAL
- ORG
- SPACE
- TITLE

ASSEMBLER STATEMENT

Operation Mnemonic

COMMON

Operand Form

No operand permitted.

Legend

None.

Notes

No label permitted.

EFFECT

Scope of COMMON Statement

A COMMON statement takes effect when it is read during assembly. A NORMAL statement causes this effect to stop. Thus, any statements following the COMMON statement and preceding the NORMAL statement (or END statement) are subject to the COMMON statement and are said to lie within its scope. When assembly begins, the NORMAL condition prevails.

Constant Address Fields

For statements within the scope of a COMMON statement, any unflagged constant address is assembled as though it had been flagged with a C to indicate an address in the common region of memory. This applies to the A and B field of an instruction statement. It also applies to address constants in the data field of a DM (Define Memory) statement. To assemble a constant address in partition from within a statement which lies within the scope of a COMMON statement, the programmer merely flags the constant with a P:

```
COMMON
S 300(5),400P(5)
```

In this example, the operand at common location 300 is subtracted from the operand at partition location 400. In other words, when the statement is assembled, AC is set ON and BC is set OFF.

Labelled Statement Reference

Each new statement label within the scope of a COMMON statement is assigned an address in the common region of memory using the common memory assignment counter. Accordingly, each time the label is referenced, the address provided by the assembler is also in the common region. This holds true whether or not the referencing statement lies within the scope of a COMMON statement.

```
NORMAL
S  A,B
```

```
COMMON
ORG 300
A DM C10
B DM C10
```

In the Subtract statement above, the operand at location 300 common is subtracted from the operand at location 310 common. Both AC and BC are turned ON because the statements labelled A and B are both within the scope of the COMMON statement.

* Address Reference

Any statements within the scope of a COMMON statement which can properly contain * as an address indicator will be assumed to have a common address.

```
COMMON
ORG 500
S  *+100,*+200
```

In the above example, the character in location 600 common is subtracted from the character in location 700 common.

ASSEMBLER STATEMENT

Operation Mnemonic

DM

Operand Form

RTS'DATA'(L,I)

Legend

| PARAMETER | RANGE | REQ/OPT | DEFAULT | DESCRIPTION |
|-----------|--------|----------|---------|---|
| R | 0-9999 | Optional | 1 | Repetition Factor |
| T | C,N,A | REQUIRED | none | Data Type |
| S | 1-9999 | Optional | Note 1 | Size of Data |
| DATA | Note 2 | Optional | none | Initialization Data |
| L | 1-100 | Optional | Note 3 | Length Specification for Label Table |
| I | 0-3 | Optional | 0 | Index Specification for Label Table |

Notes

Note 1

Default value S=1 for data types C and N.

Default value S=4 for data type A.

Examples of DM statements are shown in figure 3-1.

| SEQ# | LOCN | INSTR/DATA | OP | A/R | L I | B/S | L I | LINE | LABEL | OPCODE | OPERAND(S) | AND/OR COMMENTS |
|---|------------|------------|----|------|-----|------|-----|------|-------|--------|-----------------------|--|
| PAGE 0001 10/12/71 (4-02) PROGRAM LISTING FOR EXAMPLES OF DM STATEMENTS | | | | | | | | | | | | |
| 0000 | | | | | | | | 0002 | | DM | A | * DM EXAMPLES OF TYPE A(ADDRESS). |
| 0000 | | | | | | | | 0003 | | DM | A | * SAME AS 1A+. FOUR POS. OF MEMORY RESERVED |
| 0004 | 0000 | | | 0001 | | 0004 | | 0004 | | DM | A'0' | * FOUR ZEROS PUT INTO CORE |
| 0008 | 0300 | | | 0001 | | 0004 | | 0005 | | DM | A'300' | * ADDRESS 0300 PUT INTO CORE. |
| 0012 | 660P | | | 0001 | | 0004 | | 0006 | | DM | A'6600C' | * COMMON ADDRESS SPECIFICATION. |
| 0016 | 0660 | | | 0001 | | 0004 | | 0007 | | DM | A'660P' | * ALTERNATE WAY TO SPECIFY COMMON ADDRESS |
| 0020 | 6600 | | | 0001 | | 0004 | | 0008 | | DM | A'6600P' | * PARTITION ADDRESS SPECIFICATION. |
| 0024 | 0064 | | | 0001 | | 0004 | | 0009 | | DM | A'+40' | * ADDRESS OF CURRENT LOC. PLUS 40. |
| 0028 | 0183 | | | 0001 | | 0004 | | 0010 | | DM | A'CARD' | * ADDRESS OF LABEL CARD PUT INTO CORE |
| 0032 | | | | | | | | 0011 | | DM | | * DM EXAMPLES OF TYPE C(CHARACTER). |
| 0032 | | | | 0001 | | 0001 | | 0012 | | DM | C | * ONE POSITION OF CORE RESERVED. |
| 0033 | ' | | | 0001 | | 0001 | | 0013 | | DM | C'''' | * ONE APOSTROPHE PUT INTO CORE |
| 0034 | SIZE EQUAL | | | 0001 | | 0014 | | 0014 | | DM | C'SIZE EQUALS 14' | * SIZE IS NO. OF CHARACTERS. |
| 0048 | | | | 0001 | | 0080 | | 0015 | | DM | C80 | * EIGHTY POSITIONS OF CORE RESERVED |
| 0128 | 1 | | | 0001 | | 0004 | | 0016 | | DM | C4'1' | * ONE FOLLOWED BY THREE BLANKS PUT INTO CORE |
| 0132 | BLANK FILL | | | 0001 | | 0040 | | 0017 | | DM | C40'BLANK FILL AFTER' | * BLANKS UNTIL SIZE = 40. |
| 0172 | TYPE AN 'S | | | 0001 | | 0011 | | 0018 | | DM | C11'TYPE AN 'S'''' | * TWO APOSTROPHES TO MAKE ONE. |
| 0183 | | | | 0000 | | 0080 | | 0019 | CARD | DM | 0C80 | * NO POS. OF CORE RESERVED, BUT S=80. |
| 0183 | ABCDE | | | 0004 | | 0005 | | 0020 | | DM | *C5'ABCDE' | * FOUR SETS OF ABCDE PUT INTO CORE. |
| 0203 | ABCDE | | | 0001 | | 0005 | | 0021 | | DM | C5'ABCDE'(3) | * LENGTH IS THREE WHEN REFERENCED. |
| 0208 | | | | 0001 | | 0080 | | 0022 | C | DM | C80(1;1) | * INDEX REG. POINTS TO CHAR. IN RECORD. |
| 0288 | | | | | | | | 0023 | | DM | | * DM EXAMPLES OF TYPE N(NUMERIC). |
| 0288 | | | | 0001 | | 0001 | | 0024 | | DM | N | * ONE POSITION OF CORE RESERVED. |
| 0289 | | | | 0001 | | 0006 | | 0025 | | DM | N6 | * SIX POSITIONS OF CORE RESERVED. |
| 0295 | 365 | | | 0001 | | 0003 | | 0026 | | DM | N'365' | * 365 PUT INTO CORE. |
| 0298 | 6U | | | 0001 | | 0002 | | 0027 | | DM | N'+65' | * 6U PUT INTO CORE. SIGN MUST BE FIRST. |
| 0300 | 1 | | | 0001 | | 0004 | | 0028 | CTR | DM | N4'1' | * LEFT ZERO FILL=0001 PUT INTO CORE. |
| 0304 | 25 | | | 0004 | | 0002 | | 0029 | | DM | 4N'25' | * FOUR SETS OF 25 PUT INTO CORE. |
| 0312 | | | | | | | | 0030 | | DM | | END |

| LABEL | LOC | LNTH | INDX | MESSAGES |
|---|------|------|------|----------|
| PAGE 0001 10/12/71 (4-02) LABEL TABLE FOR EXAMPLES OF DM STATEMENTS | | | | |
| C | 0208 | 01 | 1 | UNUSED |
| CARD | 0183 | 80 | 0 | |
| CTR | 0300 | 04 | 0 | UNUSED |
| 3 LABELS | | | | |

Figure 3-1. Examples of DM Statements

Note 2

Data type governs data range:

N (Numerical) data type:

Range is limited by space available in statement for the 'DATA' field.

Data may be preceded by + or -.

Number of characters in data field (excluding + or -) may not exceed S.

C (Character) data type:

Any System Ten character is allowable.

To assemble an embedded apostrophe, use two apostrophes.

To assemble an embedded percent sign, use two percent signs.

Number of characters is limited by space available in statement for the 'DATA' field.

Number of characters to be assembled may not exceed S.

A (Address) data type:

Range, 0-9999. Usually represented symbolically.

Note 3

Default L=S if S is explicit.

If S is omitted, default L=number of characters to be assembled from data field.

If S and data field are omitted, default L=1.

EFFECT

The Define Memory statement is used to specify data fields. The assembler assigns addresses, assembles initial data into the assigned addresses, and (for each labelled DM statement) maintains associated address, length, and index parameters in the label table.

Alignment considerations

The assembler makes no attempt to align space reserved by DM statements at tens boundary addresses (as required for machine instruction statements). For a given DM statement, the assembler simply assigns the next location beyond those required by the previous statement.

Data Types C, N, A

The programmer specifies the data type for each DM statement by writing C, N, or A in the position reserved for T in his assembler statement. If he wants data assembled, he specifies the data in the data field, but in conformity to the requirements of type.

Type C

Type C (Character) data includes any System Ten character. If the programmer encodes a data field which is shorter than the field width S, the given information is left adjusted and filled with trailing blanks. To assemble an apostrophe into the data field, the programmer uses two adjacent apostrophes for every one he wishes assembled. The first and last apostrophes (which show the bounds of the data field) are not considered part of the data and are not assembled. To assemble a percent sign into the data field _ the programmer uses two adjacent percent signs for every one he wishes assembled.

Type N

Type N (Numerical) data includes digits 0-9. The numerical information can be optionally preceded by a plus (+) or a minus (-) sign. Only one such sign can be included in the data field (between the bounding apostrophes); when used, it must always be the first character in the data field. If the given information is preceded by a minus sign, bit-7 of the rightmost digit of the assembled field is turned ON. If the programmer encodes a data field which is shorter than the field width S, the given information is right-adjusted, and the field is filled with preceding zeros.

Type A

Type A (Address) DM statements are used to assemble addresses into data fields. Each assembled field will contain a four digit address. If the address refers to the common region, bit-7 of the rightmost digit is set ON. If the address refers to memory in the partition, bit-7 is set OFF.

If the programmer provides S (field width) in his DM statement, it must be 4 (for Type A). If he omits S, it is assumed to be 4.

The information the programmer provides in the data field is governed by the same rules as are used in writing addresses in machine instruction fields A and B.

The information inside the bounding apostrophes must be one of the following:

```
*
*constant
(flagged) constant
(flagged) constant constant
symbol
symbol constant
```

Field Width

S determines the field width, and R determines the number of times the field is repeated. $R \times S + \text{assembly-address}$ cannot exceed 9999. If S is omitted, and if data is given, the data determines the field width.

If both S and data are omitted, space is reserved on the basis of S=1 for C and N data types, and S=4 for A data type.

Repetition and Space Reservation

R determines the number of times the field is repeated in memory.

If R is zero, no memory is reserved, and no data are assembled for loading. If such a statement contains a label, however, the label is entered into the label table along with the associated address, length, and index register.

If R is more than 0, and if the programmer supplies data, the field is assembled R times in adjacent memory locations. If the programmer omits the data field, $R \times S$ locations are reserved but the contents of the reserved locations are unspecified.

Initial Data

The rules which govern initial data are described above under "Data Types". If the data field is omitted, no code is assembled, although the other functions of the DM statement are performed by the assembler.

Length

Length is the value which the assembler inserts in the label table as the implied length associated with a given label. If, in a machine instruction statement elsewhere in the program, a length parameter is omitted and the associated length address field (A or B) is a symbol, the length is supplied by the assembler from the corresponding label table entry.

L determines what length the assembler inserts into the label table. If L is omitted, the field width (described above) is used. If the field width exceeds 100; L should be explicitly given (not omitted) if it is to be used as described above. L should never exceed 100.

If the DM statement is unlabelled, no entry is made in the label table.

Caution

Whenever the assembler inserts an implied length into an instruction field, the length taken from the label table must be consistent with the length permitted by the operand form. For the two-length form, the length supplied from the label table cannot exceed 10, while for the one-length form, the length cannot exceed 100. If this rule is violated, the message EXCESS LENGTH is printed in the label table listing for each label where the length is too large to be assembled into a referencing instruction. After the label table is printed, the assembly is discontinued without proceeding to pass 2.

Index

Index is the value which the assembler inserts in the label table as the implied index register associated with a given label. If, in a machine instruction statement elsewhere in the program, an index specification is omitted and the associated address field (A or B) is a symbol, the index is supplied by the assembler from the corresponding label table entry.

I determines what index the assembler inserts in the label table. If I is omitted, 0 is inserted.

If the DM statement is unlabelled, no entry is made in the label table.

Implied Forms

Except for the T (type) parameter, any of the other parameters in the DM statement can be omitted:

When the data field is omitted, the bounding apostrophes should be omitted also.

When the L parameter is omitted and I is given, the separating comma should be retained.

When L is given and I is omitted, the separating comma should be omitted also.

When both L and I are omitted, the bounding parenthesis and separating comma should be omitted also.

When a parameter is omitted, the assumption which the assembler makes about the omitted parameter is as described in the preceding discussion starting with "Effect."

ASSEMBLER STATEMENT**Operation Mnemonic**

EJECT

Operand Form

No operand permitted.

Notes

No label permitted.

EFFECT

The EJECT statement advances the listing of the source program to the top of the next page.

An EJECT statement is not copied to the source program listing.

EJECT has no effect when the workstation is used as the print device.

ASSEMBLER STATEMENT

Operation Mnemonic

END

Operand Form

No operand permitted.

Legend

None.

Notes

No label permitted.

EFFECT

The END statement is taken by the assembler to be the last statement in the source program.

There must be (only) one END statement in every source program.

ASSEMBLER STATEMENT

Operation Mnemonic

EXEC

Operand Form

*
* constant
Symbol
Symbol constant
(Flagged) constant
(Flagged) constant constant

Legend

The operand specifies the address at which execution will begin.

Notes

No label permitted.

EFFECT

EXEC does not terminate the assembly process (as does the END statement).

Object

EXEC generates an S card in the object program containing the address specified by the programmer. The S card is interpreted during loading. It stops the loading process and passes control to the specified address.

Use in Overlays

If a source program contains several overlays, the last statement of each overlay must be an EXEC statement pointing to the start address for the given overlay. EXEC, used in this manner, causes loading to stop after the overlay has been loaded. Control then passes to whatever location the programmer specifies in the EXEC statement.

Resumption of Loading

The loader normally resides in partition location 50-299 throughout execution of a program. Typically, the statement ahead of each EXEC statement (for all overlays except the last) is a branch to the loader (location 60) to begin loading the next overlay.

ASSEMBLER STATEMENT

Operation Mnemonic

NORMAL

Operand Form

No operand permitted.

Legend

None.

Notes

No label permitted.

EFFECT

Scope of NORMAL Statement

A NORMAL statement takes effect when it is read during assembly. A COMMON statement causes this effect to stop. Thus, any statements following the NORMAL statement and preceding the COMMON statement (or END statement) are subject to the NORMAL statement and are said to lie within its scope. When assembly begins, the NORMAL condition prevails.

Constant Address Fields

For statements within the scope of a NORMAL statement, any unflagged constant address is assembled as though it had been flagged with a P to indicate an address in the partition region of memory. This applies to both the A and B field of an instruction statement. It also applies to address constants in the data field of a DM (Define Memory) statement. To assemble a constant address in common from within a statement which lies in the scope of a NORMAL statement, the programmer merely flags the constant with a C:

```
NORMAL
S 300(5),400C(5)
```

In this example, the operand at partition location 300 is subtracted from the operand at common location 400. In other words, when the statement is assembled AC is set OFF and BC is set ON.

Labelled Statement Reference

Each new statement label within the scope of a NORMAL statement is assigned an address in the partition region of memory using the partition memory assignment counter. Accordingly, each time the label is referenced, the address provided by the assembler is also in the partition region. This holds true whether or not the referencing statement lies within the scope of a NORMAL statement.

```
COMMON
S      A,B
```

```
NORMAL
ORG 300
A DM  C10
B DM  C10
```

In the Subtract statement above, the operand at location 300 partition is subtracted from the operand at location 310 partition. Both AC and BC are turned OFF because the statements labelled A and B are both within the scope of the NORMAL statement.

* Address Reference

Any statements within the scope of a NORMAL statement which can properly contain * as an address indicator will be assumed to have a partition address.

```
NORMAL
ORG 500
S      *+100,*+200
```

In the above example, the character in location 600 partition is subtracted from the character in location 700 partition.

ASSEMBLER STATEMENT

Operation Mnemonic

ORG

Operand Form

```
*  
* constant  
Symbol  
Symbol constant  
(Flagged) constant  
(Flagged) constant constant
```

Notes

Label permitted.

The operand may be omitted, in which case the memory assignment counter is set to the highest value yet attained by it during the current assembly. If the operand is omitted, at least 8 blank characters must lie between the operation field and the first character of the comment.

A symbol in the operand of the ORG statement pre-supposes that the symbol already has a defined value in the label table. Failure to pre-define a symbol (by using it as a statement label in an earlier statement) results in an assembly error.

If the operand is a symbol or a flagged constant, the resultant address must be in common if a COMMON statement is in effect; the address must be in partition if a NORMAL statement is in effect. The following violates this rule and constitutes an error:

```
COMMON  
  
ORG 300P
```

EFFECT

ORG sets the current memory assignment counter (common or partition) to whatever value is specified in the operand. The statement produces no object code.

Memory Assignment Counters

The assembler maintains two memory assignment counters: one for assigning addresses to statements in the scope of a COMMON statement (the common counter) and one for assigning addresses in partition (the partition counter). When the assembler is loaded, the partition counter is used; it contains the value 0000. The common counter contains the value 0000.

The assembler uses the partition counter until a COMMON statement is encountered. It then uses the common counter until a NORMAL statement is encountered, at which time it returns to using the partition counter (at the value attained before the COMMON statement). Similarly, another COMMON statement reinstates the use of the common counter (at the value attained before the NORMAL statement).

Memory Assignment Process

Each DM statement is assigned space starting with an address whose value is that of the memory assignment counter currently in use. A DM statement preempts Rxs characters (providing of course that these parameters are explicitly coded).

Each machine instruction statement is assigned 10 characters starting with the address whose value is that of the memory assignment counter after it is first adjusted, if necessary, to a tens boundary.

The other assembler commands do not preempt space and (except for the ORG statement) do not affect the memory assignment counter.

After each statement is processed, the memory assignment counter is incremented by whatever number of characters the statement preempts, and processing continues with the next statement.

The ORG statement causes a departure from this systematic process. The statement causes the current memory assignment counter (common or partition) to be set as specified in the operand field of the statement.

ASSEMBLER STATEMENT**Operation Mnemonic**

SPACE

Operand Form

A constant (range 1-99).

Notes

No label permitted.

EFFECT

The SPACE statement causes the generation of blank lines on the listing of the source program. The operand determines the number of blank lines.

If the operand equals or exceeds the number of blank lines remaining on the current page, the paper merely advances to the top of the next page.

If the operand is less than the number of blank lines remaining on the current page, the number of blank lines specified by the operand are generated.

If the operand is omitted or 0, one blank line is generated.

A SPACE statement is not copied to the source program listing.

ASSEMBLER STATEMENT

Operation Mnemonic

TITLE

Operand Form

'title text enclosed by apostrophes'

Legend

Use two apostrophes for each apostrophe to be printed.

Notes

Label not permitted.

The TITLE statement, if used, must precede all other statements.

Each assembly can process only one TITLE statement.

EFFECT

The title text is stored in a buffer internal to the assembler. Each time the assembler begins a new page in the listing, it prints a line containing the title text.

Section 4
ASSEMBLER OUTPUT

INTRODUCTION
LISTING OUTPUT
ERROR MESSAGES
OBJECT DECK FORMATS

ASSEMBLER OUTPUT

INTRODUCTION

This section describes the output generated by the assembler. First, the listing output is described, after which the object deck formats are described. Figure 4-1 is an example of the program listing and output generated by an error-free Assembler I source program, with the optional cross-reference listing included.

LISTING OUTPUT

The listing output is generated on whatever device is specified as the listing device. (See Appendix A: Assembler Deck Parameters.) If the printer is used as the listing device, it will skip to the top of a new page before printing the first line of output.

The listing output has four types of listings.

1. Error Statement Listing (generated only if errors are encountered).
2. Label Table Listing (created in all cases).
3. Program Listing (produced only if no errors are encountered).
4. Symbol Cross-Reference Listing (output if requested by the user, and only if no errors are encountered). This feature is provided only with the disc version of the assembler.

The program title (if specified in the TITLE assembler command statement) and the date (if entered via the DATE assembler parameter statement) appear in the headings on each page of all of the above documents.

ASSEMBLER OUTPUT

Error Statement Listing

During pass 1, each statement is checked for errors and each erroneous statement is listed in the Error Statement Listing. The listing consists of these fields:

| <u>Field</u> <u>Heading</u> | <u>Contents</u> |
|---|---|
| SEQ | The line number assigned to this statement by the assembler, according to its actual position in the source code assembled. |
| LOCN | The actual location in core of the erroneous machine instruction or data field. The notation **** in this field indicates an out-of-bounds address (above 9999), or an attempt to set the memory assignment counter to an erroneous value (via an ORG statement). |
| 1...10...20.... (Headings on every tenth column, 1-80.) | The erroneous source statement. Within the source statement, each error is flagged with an asterisk on the following line. Where an asterisk appears to the immediate left of a source statement label, this means that the label was previously defined in another source statement. The SEQ and LOCN fields refer only to the current error statement, and not to any previous statement having the same label. |

If there are any errors, the assembly will be discontinued after the label table has been printed (pass 2 will not occur). An example of an error statement listing with the associated label table appears in figure 4-2.

PAGE 0001 10/12/71 (4-02) PROGRAM LISTING FOR EXAMPLE OF AN ASSEMBLER ONE EXECUTABLE PROGRAM

| SEQ. | LOCN | INSTR/DATA | OP | A/R | L | I | B/S | L | I | LINE | LABEL | OPCODE | OPERAND(S) | AND/OR COMMENTS |
|------|------|------------|----|------|---|---|------|---|---|------|--------|--------|----------------|--|
| 0000 | | | | | | | | | | 0002 | * | | | THIS EXAMPLE CALCULATES A BASEBALL PITCHER'S EARNED RUN AVERAGE. |
| 0000 | | | | | | | | | | 0003 | * | | | THE INPUT IS FROM CARDS WITH UP TO TEN SETS OF DATA ON A CARD. |
| 0000 | | | | | | | | | | 0004 | * | | | THE OUTPUT IS TO THE WORKSTATION. |
| 0000 | | | | | | | | | | 0005 | | ORG | 11 | |
| 0011 | | | | 0011 | | | 0004 | | | 0006 | XR1 | DM | N4 | |
| 0015 | | | | 5000 | | | | | | 0008 | | ORG | 5000 | |
| 5000 | | QJ2J040011 | 13 | 5250 | 1 | 0 | 0011 | 4 | 0 | 0309 | INIT | FN | ZERO, | XR1 * ZEROS TO REGISTER |
| 5010 | | 1524710080 | 00 | 5287 | 1 | 0 | 0080 | 1 | 0 | 0010 | | R | INPUT(1), | 80(1) * READ A CARD FROM THE READER |
| 5020 | | 55RU800000 | 11 | 5258 | 3 | 0 | 0000 | 0 | 0 | 0011 | | BC | BLANKS(3) | * END JOB IF UNIT SEPERATOR. |
| 5030 | | PUR473U258 | 14 | 5287 | 0 | 1 | 5258 | 3 | 0 | 0012 | LOOP | C | IP, | BLANKS * IS REST OF CARD BLANK? |
| 5040 | | R5PP000000 | 11 | 5000 | 2 | 0 | 0000 | 0 | 0 | 0013 | | BC | INIT(2) | * YES, READ NEXT CARD. |
| 5050 | | QJ2J035373 | 13 | 5250 | 1 | 0 | 5373 | 3 | 0 | 0014 | | FN | ZERO, | QUOT+6(3) * CLEAR OUT LAST 3 POS. OF DIVIDEND |
| 5060 | | SJ2X73U376 | 13 | 5287 | 3 | 1 | 5376 | 3 | 0 | 0015 | | FN | IP, | DIVSOR(3) * PUT IN LEFT MOST 3 POSITIONS. |
| 5070 | | 1UR5135376 | 06 | 5251 | 1 | 0 | 5376 | 3 | 0 | 0016 | | M | THREE, | DIVSOR(3) * RESULT IS FOUR POSITIONS. |
| 5080 | | 1J2904U376 | 04 | 5290 | 1 | 1 | 5376 | 4 | 0 | 0017 | | A | FIP, | DIVSOR * ADD TO RESULT |
| 5090 | | SJ2Y14U367 | 13 | 5291 | 3 | 1 | 5367 | 4 | 0 | 0018 | | FN | RUNS, | QUOT(4) * LEADING ZERO ASSURED. |
| 5100 | | 2UR5235368 | 06 | 5252 | 2 | 0 | 5368 | 3 | 0 | 0019 | | M | TWUSEV, | QUOT+1(3) * DIVIDEND IS 0, 5 POSITIONS, AND |
| 5110 | | 4U3W655367 | 05 | 5376 | 4 | 0 | 5367 | 5 | 0 | 0020 | | D | DIVSOR, | QUOT * 3 TRAILING ZEROS FOR 3 DECIMAL PLACES |
| 5120 | | T5RR000000 | 11 | 5220 | 4 | 0 | 0000 | 0 | 0 | 0021 | | BC | ERRWRT(4) | * CHECK FOR OVERFLOW. |
| 5130 | | 1J25655367 | 04 | 5256 | 1 | 0 | 5367 | 5 | 0 | 0022 | | A | FIVE, | QUOT * ROUND OFF RESULT |
| 5140 | | P526165380 | 08 | 5261 | 0 | 0 | 5380 | 6 | 0 | 0023 | | MC | EDITDM, | ERA * MOVE IN EDIT MASK |
| 5150 | | PU36745380 | 12 | 5367 | 0 | 0 | 5380 | 4 | 0 | 0024 | | E | QUOT(4), | ERA * PUT ANSWER IN WITH EDITING |
| 5160 | | P538065273 | 08 | 5380 | 0 | 0 | 5273 | 6 | 0 | 0025 | | MC | ERA, | MSG+6 * ANSWER TO MESSAGE |
| 5170 | | 052J730001 | 01 | 5257 | 0 | 0 | 0001 | 3 | 0 | 0026 | | W | CR(0), | 1(3) * CARRIAGE RETURN ON WORKSTATION. |
| 5180 | | 052V710012 | 01 | 5267 | 0 | 0 | 0012 | 1 | 0 | 0027 | | W | MSG(0), | 12(1) * WRITE ANSWER ON WORKSTATION. |
| 5190 | | 1J25440011 | 04 | 5254 | 1 | 0 | 0011 | 4 | 0 | 0028 | | A | EIGHT, | XR1 * TO GET NEXT SET OF DATA |
| 5200 | | PJRS420013 | 14 | 5254 | 0 | 0 | 0013 | 2 | 0 | 0029 | | C | EIGHT(2), | XR1+2 * HAS THERE BEEN 10 SETS OF DATA? |
| 5210 | | R5PP055030 | 11 | 5000 | 2 | 0 | 5030 | 5 | 0 | 0030 | | BC | INIT(2), | LOOP(5) * YES, READ CARD; NO, PROCESS DATA. |
| 5220 | | 052J730001 | 01 | 5257 | 0 | 0 | 0001 | 3 | 0 | 0031 | ERRWRT | W | CR(0), | 1(3) * CARRIAGE RETURN ON WORKSTATION. |
| 5230 | | 052W910008 | 01 | 5279 | 0 | 0 | 0008 | 1 | 0 | 0032 | | W | ERRMSG(0), | 8(1) * WRITE OVERFLOW MESSAGE ON WORKSTATION |
| 5240 | | USRU800000 | 11 | 5258 | 5 | 0 | 0000 | 0 | 0 | 0033 | | BC | BLANKS(5) | * END JOB |
| 5250 | | | | 0001 | | | 0001 | | | 0035 | ZERO | DM | N*0' | |
| 5251 | | | | 0001 | | | 0001 | | | 0036 | THREE | DM | N*3' | |
| 5252 | | | | 0001 | | | 0002 | | | 0037 | TWUSEV | DM | N*27' | |
| 5254 | | | | 0001 | | | 0002 | | | 0038 | EIGHT | DM | N*80'(1) | |
| 5256 | | | | 0001 | | | 0001 | | | 0039 | FIVE | DM | N*5' | |
| 5257 | | | | 0001 | | | 0001 | | | 0040 | CR | DM | C*1' | |
| 5258 | | | | 0001 | | | 0003 | | | 0041 | BLANKS | DM | C*3' | |
| 5261 | | 0-00- | | 0001 | | | 0006 | | | 0042 | EDITDM | DM | C*0-00+' | |
| 5267 | | ERA = 00.0 | | 0001 | | | 0012 | | | 0043 | MSG | DM | C*ERA = 00.00' | |
| 5279 | | OVERFLOW | | 0001 | | | 0008 | | | 0044 | ERRMSG | DM | C*OVERFLOW' | |
| 5287 | | | | 0000 | | | 0080 | | | 0045 | INPUT | DM | OC80 | |
| 5287 | | | | 0001 | | | 0003 | | | 0046 | IP | DM | N3(,1) | * INNINGS PITCHED |
| 5290 | | | | 0001 | | | 0001 | | | 0047 | FIP | DM | N1(,1) | * FRACTIONAL INNINGS PITCHED=0,1,OR 2. |
| 5291 | | | | 0001 | | | 0003 | | | 0048 | RUNS | DM | N3(,1) | * EARNED RUNS ALLOWED |
| 5294 | | | | 0001 | | | 0073 | | | 0049 | FR | DM | C*3 | * REMAINDER OF CARD IMAGE. |
| 5367 | | | | 0001 | | | 0005 | | | 0050 | QUOT | DM | N5 | * ANSWER OF DIVIDE WILL BE 5 POSITIONS. |
| 5372 | | | | 0001 | | | 0004 | | | 0051 | DM | | N4 | * DIVIDEND WILL INCLUDE THESE POSITIONS |
| 5376 | | | | 0001 | | | 0004 | | | 0052 | DIVSOR | DM | N4 | |
| 5380 | | | | 0001 | | | 0006 | | | 0053 | ERA | DM | C6 | * EARNED RUN AVERAGE |
| 5386 | | | | 5000 | | | | | | 0055 | EXEC | INIT | | |
| 5386 | | | | | | | | | | 0056 | END | | | |

PAGE 0001 10/12/71 (4-02) LABEL TABLE FOR EXAMPLE OF AN ASSEMBLER ONE EXECUTABLE PROGRAM

| LABEL | LOC | LNTH | INDX | MESSAGES |
|-----------|------|------|------|----------|
| BLANKS | 5258 | 03 | 0 | |
| CR | 5257 | 01 | 0 | |
| DIVSOR | 5376 | 04 | 0 | |
| EDITDM | 5261 | 06 | 0 | |
| EIGHT | 5254 | 01 | 0 | |
| ERA | 5380 | 06 | 0 | |
| ERRMSG | 5279 | 08 | 0 | |
| ERRWRT | 5220 | 10 | 0 | |
| FIP | 5290 | 01 | 1 | |
| FIVE | 5256 | 01 | 0 | |
| INIT | 5000 | 10 | 0 | |
| INPUT | 5287 | 80 | 0 | |
| IP | 5287 | 03 | 1 | |
| LOOP | 5030 | 10 | 0 | |
| MSG | 5267 | 12 | 0 | |
| QUOT | 5367 | 05 | 0 | |
| RUNS | 5291 | 03 | 1 | |
| THREE | 5251 | 01 | 0 | |
| TWUSEV | 5252 | 02 | 0 | |
| XR1 | 0011 | 04 | 0 | |
| ZERO | 5250 | 01 | 0 | |
| 21 LABELS | | | | |

PAGE 0001 10/12/71 CROSS REFERENCE LISTING

| DEF. | SYMBOL | REFERENCES |
|------|--------|-------------------------------|
| 0040 | BLANKS | 0010 0011 0032 |
| 0039 | CR | 0025 0030 |
| 0051 | DIVSOR | 0014 0015 0016 0019 |
| 0041 | EDITDM | 0022 |
| 0037 | EIGHT | 0027 0028 |
| 0052 | ERA | 0022 0023 0024 |
| 0043 | ERRMSG | 0031 |
| 0030 | ERRWRT | 0020 |
| 0046 | FIP | 0016 |
| 0038 | FIVE | 0021 |
| 0008 | INIT | 0012 0029 0054 |
| 0044 | INPUT | 0009 |
| 0045 | IP | 0011 0014 |
| 0011 | LOOP | 0029 |
| 0042 | MSG | 0024 0026 |
| 0049 | QUOT | 0013 0017 0018 0019 0021 0023 |
| 0047 | RUNS | 0017 |
| 0035 | THREE | 0015 |
| 0036 | TWUSEV | 0018 |
| 0005 | XR1 | 0008 0027 0028 |
| 0034 | ZERO | 0008 0013 |

Figure 4-1. Error-Free Assembler Output

ASSEMBLER OUTPUT

```

PAGE 0001 10/12/71 (4-02) ASSEMBLY ERRORS FOR ASSEMBLER I ERROR TEST CARDS.
SEQ LOCN 1.....10.....20.....30.....40.....50.....60.....70.....80
0002 0000 DM01 DM NA'4444' * REPETITION OR SIZE NOT NUMERIC.
**** *
0003 0004 DM02 DM BA'7777' * REPETITION FACTOR NOT NUMERIC.
**** **
0004 0008 DM03 DM ND'333' * SIZE FACTOR NOT NUMERIC.
**** *
0005 0011 DM04 DM N'07=24=70' * NON-NUMERICS IN NUMERIC TYPE.
**** * *
0006 0017 DM05 DM AS'4444' * SIZE MUST BE 4.
**** *
0007 0022 DM06 DM D'1010' * TYPE IS WRONG.
**** *
0008 0026 DM07 DM 6'4220' * NO TYPE SPECIFIED
**** * *
0009 0026 DM08 DM C80' '(,4) * INDEX MUST BE 0-3.
**** *
0010 0106 DM09 DM C4'AB * NO ENDING APOS.
**** *****
0011 0106 DM10 DM C4ABCD' * NO STARTING APOS.
**** *****
0012 0110 DM11 DM A'BC ' * SPACES IN DATA
**** **
0013 0114 DM12 DM C25'THE SIZE IS NOT BIG ENOUGH FOR THIS CONSTANT'
**** *****
0014 0139 DM13 DM ON0'7777' * SIZE CAN NOT BE ZERO
**** *
0015 0139 DM14 DM CC6'HELLO' * AN EXTRA C
**** *
0016 0144 DM15 DM N'4=-' * MINUS SIGN MUST BE FIRST
**** *
0017 0150 DM16 C' ' * NO OPERATION CODE
**** **
0018 0160 DM16 A DM03,DM08+76 * DUPL. LABEL, B OPERAND EXCESS LENGTH.
**** *
0019 0170 A01 A DM01,DM03(J) * NON-NUMERIC LB
**** *
0020 0180 MC01 MC DM02,DM04(6,1) * SHOULD BE (,1)-LB NOT ALLOWED.
**** *
0021 0190 BC EOF(3),ERMSG(4) * MORE THAN 8 SPACES BEFORE OPERAND A.
**** **
0022 0200 MC01 MC (20),DM05 * DUPL. LABEL, NO OPERAND A
**** *****
0023 0210 RD MCO1(5) * INVALID OPERATION CODE
**** *
0024 0220 BC MCO1(2), * NO OPERAND B.
**** *
0025 0230 LABEL MC ONE,TWO * LABEL MUST START IN COL. 1.
**** *
0026 0240 TOOLONG A 12,45 * LABEL IS TOO LONG
**** *
0027 0250 DM A'TOOLONG' * LABEL TOO LONG.
**** *
0028 0260 A DM06,TOOLONG * LABEL TOO LONG
**** *
0029 0270 12 DM N'12' * LABEL MAY NOT START WITH A NUMBER.
**** *
0030 0272 ORG **SYMBOL * CONSTANT MUST BE USED.
**** *
0031 *** EOJ END * NO LABEL ALLOWED, BUT TAKEN AS END.
**** *

```

```

PAGE 0001 10/12/71 (4-02) LABEL TABLE FOR ASSEMBLER I ERROR TEST CARDS.
LABEL LOC LNTH INDX MESSAGES
A01 0170 10 0 UNUSED
BC 0012 00 0 UNDEFINED
DM01 0000 04 0
DM02 0004 04 0
DM03 0008 03 0
DM04 0011 06 0
DM05 0017 05 0 UNUSED
DM06 0022 04 0
DM07 0026 ** 0 UNUSED
DM08 0026 80 EXCESS LENGTH
DM09 0106 04 0 UNUSED
DM10 0106 04 0 UNUSED
DM11 0110 04 0 UNUSED
DM12 0114 25 0 UNUSED
DM13 0139 04 0 UNUSED
DM14 0139 05 0 UNUSED
DM15 0144 01 0 UNUSED
DM16 0160 10 0 UNUSED
MC 0230 10 0 UNUSED
MC01 0200 10 0
20 LABELS
ERRORS DETECTED

```

Figure 4-2. Error Statement Listing

Label Table Listing

For every label in the source program, the label table presents the following information in the fields indicated:

| <u>Field Heading</u> | <u>Contents</u> |
|--------------------------|--|
| LABEL | The symbol used as the statement or data field label. (These symbols are listed in alphabetic order.) |
| LOC | The location in core assigned to the label. If label is undefined, line number of undefined statement. |
| LNTH | The length attribute associated with the label. (** indicates a length greater than 100). |
| INDX | The index register attribute associated with the label. |
| MESSAGES | Any messages regarding the label symbol, such as UNUSED (to indicate that this symbol is used as a label but is not referenced as an operand by any statement) or UNDEFINED (to indicate that the symbol is referenced as an operand but does not appear as a label in any statement.) When a symbol is listed as UNDEFINED, the <u>line</u> number (assigned by the assembler) of the <u>last</u> record containing the symbol as an operand will appear under the LOC heading (in place of a core location). The message EXCESS LENGTH is printed for each label where the length is too large to be assembled into a referencing instruction (see <u>Caution</u> under DM statement effect in Section 3). |

At the end of the label table listing, the message "nnn LABELS" is printed, where nnn is the number of labels in the source program.

ASSEMBLER OUTPUT

Program Listing

The program listing is created only if no errors are detected during pass 1. The program listing is generated during pass 2, and consists of a line-by-line listing of the source program. It contains the following information:

| <u>Field</u> <u>Heading</u> | <u>Contents</u> |
|--------------------------------|---|
| SEQ. | Sequence number (if any) copied from positions 77 through 80 of each record assembled. |
| LOCN. | The actual beginning location in core of the machine instruction or data. |
| INSTR/DATA | The machine language representation of the machine instruction, or in the case of DM statements, up to the first ten characters within the apostrophes (blank or zero fill not included). |
| OP | The function code for machine instructions, in hexadecimal representation. |
| A/R | For machine instructions, this column contains the address of the A-operand; for DM statements, this column contains the repetition factor. |
| L | The contents of the LA field of the machine instruction. |
| I | The contents of the IA field of the machine instruction. |
| B/S | For machine instructions, this field contains the address of the B-operand; for DM statements, this field contains the size factor. |
| L | The contents of the LB field of the machine instruction. |
| I | The contents of the IB field of the machine instruction. |
| LINE | The line number assigned to this statement by the assembler, according to its actual position in the source code assembled. |
| LABEL | The statement label as in the source statement. |
| OPCODE | The statement operation code, as in the source statement. |

| | |
|------------|--|
| OPERAND(S) | The statement operand(s) as in the source statement. |
| COMMENTS | Any user comments associated with the statement. |

Symbol Cross-Reference Listing

The optional symbol cross-reference listing is requested with the XREF=YES assembler parameter statement. This option is available only with the disc version of the assembler. The listing contains the following information for each label encountered:

| <u>Field</u> <u>Heading</u> | <u>Contents</u> |
|--------------------------------|--|
| DEF | The number of the statement line where the label was defined. (This is the line number assigned to the statement by the assembler, according to its actual position in the source code assembled.) |
| SYMBOL | The symbol used as the label. |
| REFERENCES | The line number of each statement referencing the label symbol as an operand. |

ASSEMBLER OUTPUT

ERROR MESSAGES

To indicate errors or significant events encountered during assembly, various messages are output. These are described below.

| Message | Meaning and Correction | Where Output |
|-------------------------------------|--|---|
| A) ASMID :END OF ASSEMBLY | The assembly is complete. If a cross-reference listing was requested, this will follow. Check all listing output, prepare to load and execute object program. | Console print-out. (The console can be a Workstation or a Communications Terminal). (Output only on disc version of the assembler.) |
| ERRORS DETECTED | At least one error occurred and will be noted in the error statement listing, or a source statement referenced an undefined label. Check error statement and label table listings, correct errors, and reassemble program. | End of Label Table Listing. |
| NO END STNT | An END statement was not included in the source program. Append an END card and reassemble program. | End of Error Message Listing. |
| SYMBOL TABLE FULL | All storage allocated for the symbol (label) table has been used. Change columns 12-19 of the first card in the assembler object deck to increase the boundaries of the label table in common core and reassemble the program. | Label Table Listing |
| T) ASMID :DISC ERROR AT (adr) | An error occurred during reading of the source input from disc file, or writing of object output onto disc file. (This could include the disc pool or file label). The address where the error occurred is indicated by adr. Control returns to the DMF conversational loader, and the message ENTER PROGRAM NAME is output on the console. If the error occurred during reading of the source file, re-file the source program. If the error was encountered during writing of the object output, change the output file designation. If the pool or file label could not be read or written correctly, consult the <u>Disc Management Facility User's Reference Manual</u> . | Console print-out (Output with disc version only). |
| T) ASMID :OUTPUT FILE NOT NULL | The file designated for object output was not a null file. Control returns to the DMF conversational loader, and the message ENTER PROGRAM NAME is output on the console. Change the output file to a null file, or select for output another file that is already null. | Console printout (Output with disc version only). |
| T) ASMID :NO EXISTING pppppp | The pool designated pppppp cannot be located. Control returns to the DMF conversational loader and the message ENTER PROGRAM NAME is output on the console. If the pool does not exist, create it. If it does exist, check to see that it is referenced by its correct name. | Console printout. (Output with disc version only). |
| T) ASMID :NO EXISTING pppppp.ffffff | The file designated fffffff in pool pppppp cannot be located. Control returns to the DMF conversational loader and the message ENTER PROGRAM NAME is output on the console. If the file does not exist, create it. If it does exist, check to see that it is referenced by its correct name and is stored within the correct pool. | Console printout. (Output with disc version only). |

OBJECT DECK FORMATS

If no errors are detected in pass 1, the source deck is considered to be free of errors and the assembler proceeds to pass 2 and generates the object deck on whatever output device is specified as the object device (See Appendix A: Assembler Deck Parameters). (If errors are detected, no object deck is generated.) The object deck is serialized in columns 77-80. Column 1 of each object card is an alphabetic character showing card type.

T Card

Format

| | |
|----------|---|
| Column 1 | T |
| 2-3 | Number of characters to be loaded (from card columns 8-70). |
| 4-7 | Address into which first character (column 8) will be loaded. |
| 8-70 | Object code to be loaded. |
| 71-76 | Program Identification (copied from first source card). |
| 77-80 | Card sequence number. |

Note

The T card is the only format used for loadable object code.

S Card

Format

| | |
|----------|---|
| Column 1 | S |
| 2-5 | Start Address |
| 71-76 | Program Identification (copied from first source card). |
| 77-80 | Card sequence number. |

Note

The S card is generated by the EXEC command.

The S card causes the loader to cease loading and to pass control to the loaded program at the start address.

Section 5
USING THE ASSEMBLER

ASSEMBLER I (CARD VERSION)
ASSEMBLER I (DISC VERSION)

USING THE ASSEMBLER

The procedures for installing Assembler I and using it to assemble source programs are described in this section. These procedures for the card version and disc version of the assembler differ, and are therefore discussed separately.

ASSEMBLER I (CARD VERSION)

The user prepares the card version of the assembler by arranging a deck that consists of any System Ten object card loader, the pass 1 assembler deck (optionally including cards containing assembler parameter statements), the user's source programs, and the pass 2 assembler deck. Using a command entered on the Workstation or Communications Terminal, he loads this combined deck through the card reader, and the assembly process begins. Detailed directions follow.

Fixed Parameters

Assembler I (Card Version) is furnished to the customer as an object card deck. The first card in this deck (identified as 0001 in columns 77 through 80) contains the fixed assembler parameters used to specify various input/output options. Typically, in a card system, the fixed parameters are set when a System Ten is installed and are rarely changed. However, when changes are necessary, they are implemented by substituting for the first card another card containing new parameters.

The fixed parameters are contained in columns 8 through 39 of the first card.

USING THE ASSEMBLER

The entries in the fixed parameter card columns are as follows:

| <u>Card Column</u> | <u>Entry</u> |
|--------------------|---|
| 8 | Source Device Number (usually IOC Device 1, the card reader). |
| 9 | Listing Device Number (usually IOC Device 2, the line printer). |
| 10 | Object Device Number (usually IOC Device 4, the card punch). |
| 11 | Not used. |
| 12-15 | Lower bound of Label Table in Common Core. (Rightmost character must be an alphabetic character P to Y to indicate a negative number, and thus a Common Core address.) This address cannot be lower than 300. |
| 16-19 | Upper bound of Label Table in Common Core. (Rightmost character must be an alphabetic character P to Y to indicate a negative number, and thus a Common Core address.) |
| 20-23 | Object card trailer control characters. The assembler sends these characters (via the Write Control Mode) to the object device after each object card is generated. |
| 24-27 | Maximum number of lines that can be printed on each page of listing output. |
| 28-29 | Not used. |
| 30-37 | Default date to go on assembly listing (if no DATE= assembly parameter statement is entered). (This entry is originally blank.) |

| | |
|-------|---|
| 38 | Default disposition of debug cards if no DEBUG= assembly parameter statement is entered). |
| | <p>\$ = Ignore source cards containing \$ in column 1. (This is the entry originally in column 38.)</p> <p>* = Assemble source cards containing \$ in column 1.</p> |
| 39 | Default disposition of Common object text. |
| | <p>Y = Punch Common object text. (This entry is originally in this column.)</p> <p>9 = Assemble Common object code, but suppress it from the object deck.</p> |
| 40-46 | Constants used by the assembler. |
| 47-70 | Not used. |
| 71-76 | Program identification. |
| 77-80 | Card sequence number 0001. |

Variable Parameters

The variable assembler parameters vary from job to job. They are used to declare the following options which override the corresponding default entries in the fixed parameter card:

- . The date to appear in the listing output.
- . The disposition of Debug Cards (ignore or assemble).
- . The disposition of Common text object cards (punch or no-punch).

The variable parameters are entered through assembler parameter statements punched on cards and placed in the assembler object deck, behind the first S-card but before the first Unit Separator card (punched 11-7-8-9 in column 1). After the portion of the assembler ahead of the S-card is loaded, control passes to it and the variable parameter cards are read. As each card is processed, the corresponding fixed-parameter default entry is overwritten in memory as indicated below. If, for any of these fixed parameters, no corresponding variable parameter card is processed, that fixed entry remains the same.

USING THE ASSEMBLER

| Assembler Parameter Statement | Parameter Card Column Affected | Value Inserted | Effect on Assembly |
|-------------------------------------|--------------------------------------|-------------------|---|
| DEBUG=YES | 34 | * | Debug cards are assembled. |
| DEBUG=NO | 34 | \$ | Debug cards are ignored. |
| COMTXT=YES | 35 | Y | Common object text is punched. |
| COMTXT=NO | 35 | 9 | Common object text is not punched. |
| DATE=mm/dd/yy | 26-33 | mm/dd/yy | Date specified is printed on listing, where mm = month, dd = date, yy = year. |

Any assembler parameter statement can be cancelled by a succeeding statement. For example, DEBUG=YES followed by DEBUG=NO leaves the assembler parameter corresponding to DEBUG=NO in memory.

Assembling Source Programs

With the card assembler, various devices can be used for input of the user source program and output of listings and the object program. In the following directions, the most typical device assignments are assumed, as follows:

| Device | IOC No. Assigned | Function |
|--------------|---------------------|--|
| Workstation | 0 | Initiate loading. |
| Card Reader | 1 | Installation of assembler and input of source program. |
| Line Printer | 2 | Output of listings. |
| Card Punch | 4 | Output of object program. |

Standard device number assignments are shown in appendix G.

The first card (sequence number 0001) of the assembler object deck must contain the proper parameters to designate the card reader, line printer, and card punch IOC numbers as the source, listing, and object devices, respectively.

To prepare for assembly, put the following in the card reader's input hopper in the order shown (see figure 5-1):

1. The object card loader.
2. The Pass 1 assembler deck (including embedded parameter cards).
3. The source deck to be assembled.
4. The Pass 2 assembler deck.

Press the LOAD and LOCAL buttons on the Workstation simultaneously. Press the Workstation's ONLINE button and enter the bootstrap instruction:

1001010290

This will initiate loading and assembly.

During Pass 1, the source deck is checked for errors, and the Label Table is constructed. Near the end of Pass 1, the first few cards of the Pass 2 assembler deck are read, and the Label Table is sorted into alphabetical order.

If errors are detected, the assembly process ends with the completion of Pass 1, and the output is as follows:

1. A listing of all erroneous statements (the errors are flagged with asterisks under the errors),
2. the label table listing, with label error messages (if any),
3. the message "ERRORS DETECTED".

If no errors are detected, then the assembly process continues with Pass 2, and the only output from Pass 1 is the label table listing. If no erroneous statements appear ahead of the label table listing, the source deck should again be put in the card reader's input hopper immediately behind the remainder of the Pass 2 assembler deck. During Pass 2, the object code is assembled and punched, and the assembly listing is printed.

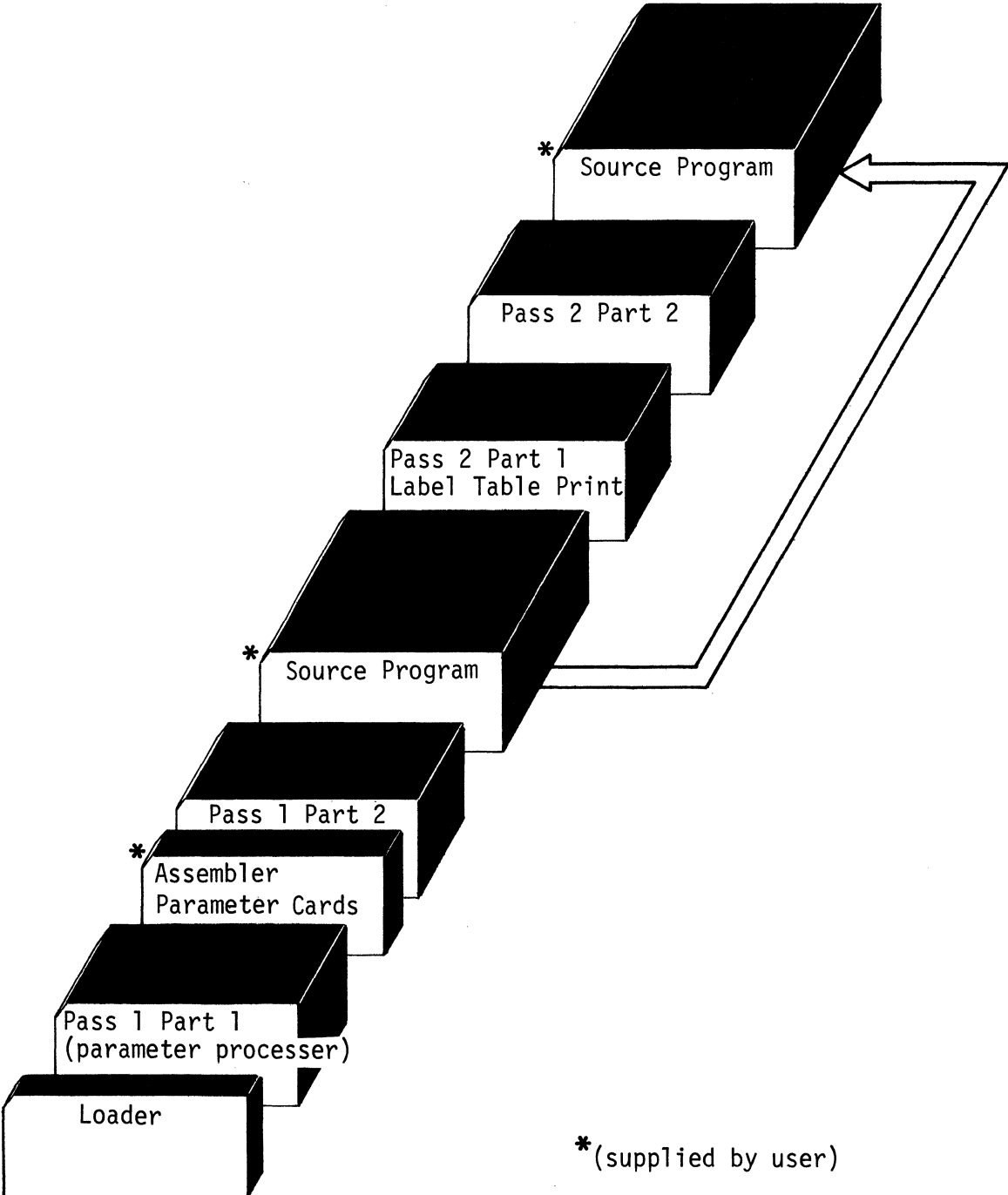


Figure 5-1. Assembler I Job Deck

ASSEMBLER I (DISC VERSION)

The disc-resident assembler operates under the Disc Management Facility (DMF). The operation differs from that of the card version in two respects:

1. The (optional) assembler parameter statements are entered via the COMMAND logical unit (_LRDR). The device address of this logical unit is established by the ASSIGN function of DMF MAINT. The device address is originally zero if no prior assignment has been made.
2. The immediate source input must be a DMF disc file.

The operator controls the assembly process from the Workstation or Communications Terminal keyboard. At his command, the FILE/REFILE program (part of DMF) reads his source deck into a DMF disc file of his choosing. At his command, Assembler I enters memory from disc to process any assembler parameter statements he enters in the keyboard. He follows his last parameter statement with a null statement generated by pressing any CTL key (on the Workstation) or entering control p control / (on the Communications Terminal). The assembly proper begins. Using the disc source file as input, the assembly is performed, the listing is printed on the listing device, and the object deck is generated on the object device (which may be a DMF file specified by the OBJECT= parameter card).

The source deck passes through the card reader only once (instead of twice as with the card version). Once the assembler is installed, it is never again read through the card reader. Instead, it enters memory from the disc file where it resides permanently. This eliminates the possibility of dropping the assembler deck and simplifies card handling.

Fixed Parameters

Assembler I (Disc Version), like the card version of the assembler, is also furnished to the user as an object card deck. The first card in this deck contains the fixed assembler parameters used to specify input/output options, but the purposes of the entries in this card differ somewhat from those in the card version of the assembler. These entries are as follows:

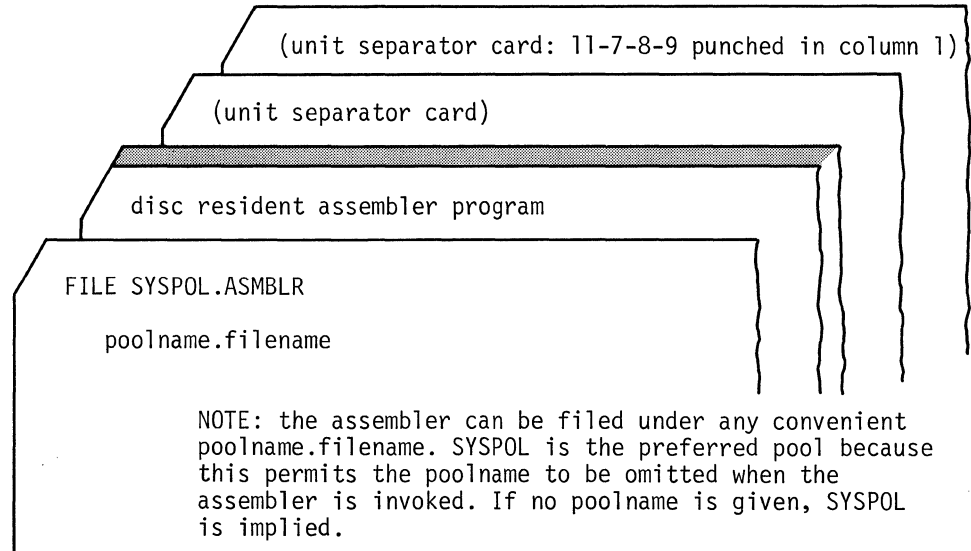
| <u>Card Column</u> | <u>Entry</u> |
|--------------------|---|
| 8-10 | Ignored. |
| 11 | Default disposition of symbol cross-reference listing (when no XREF= assembly parameter statement is used). |
| | Y = Create listing. |
| | N = Suppress listing. (Original entry.) |

USING THE ASSEMBLER

| <u>Card Column</u> | <u>Entry</u> |
|--------------------|--|
| 12-15 | Lower bound of Label Table in Common Core. (Rightmost character must be an alphabetic character P to Y to indicate a negative number, and thus a Common Core address.) |
| 16-19 | Upper bound of Label Table in Common Core. (Rightmost character must be an alphabetic character P to Y to indicate a negative number and thus, a Common Core address.) |
| 20-23 | Object card trailer control characters. The assembler sends these characters (via the Write Control Mode) to the object device after each object card is generated. |
| 24-27 | Maximum number of lines that can be printed on each page of listing output. |
| 28-29 | Not used. |
| 30-37 | Default date to go on assembly listing (if no DATE= assembly parameter statement is entered). Typically, this entry is blank. |
| 38 | Default disposition of debug cards (if no DEBUG= assembly parameter statement is entered). \$ = Ignore source cards containing \$ in column 1. (Original entry.) * = Assemble source cards containing \$ in column 1. |
| 39 | Default disposition of Common object text. Y = Write Common object text in output. (Typical entry.) 9 = Assemble Common object code, but suppress it from the object deck. |
| 40-46 | Constants used by the assembler. |
| 47-70 | Not used. |
| 71-76 | Program identification. |
| 77-80 | Card sequence number 0001. |

Installing The Assembler

Before installing the disc assembler, examine the first card (0001) of the assembler object deck to determine that it contains the fixed parameters desired. After making any necessary changes, arrange the assembler deck with the necessary DMF cards as follows:



Then proceed as follows, interacting with either the Workstation or Communications Terminal console:

Operator: Place input deck in card reader. Get load condition on console. Depress ENTER key (on Workstation) or control / key (on Communications Terminal).

Console: Prints: A) ENTER PROGRAM NAME

Operator: Type: FILE or REFILE. (Use FILE if poolname.filename entry on first input card identifies a new file. Use REFILE if the identified file already exists.)

Depress: ENTER key (on Workstation) or control / key (on Communications Terminal).

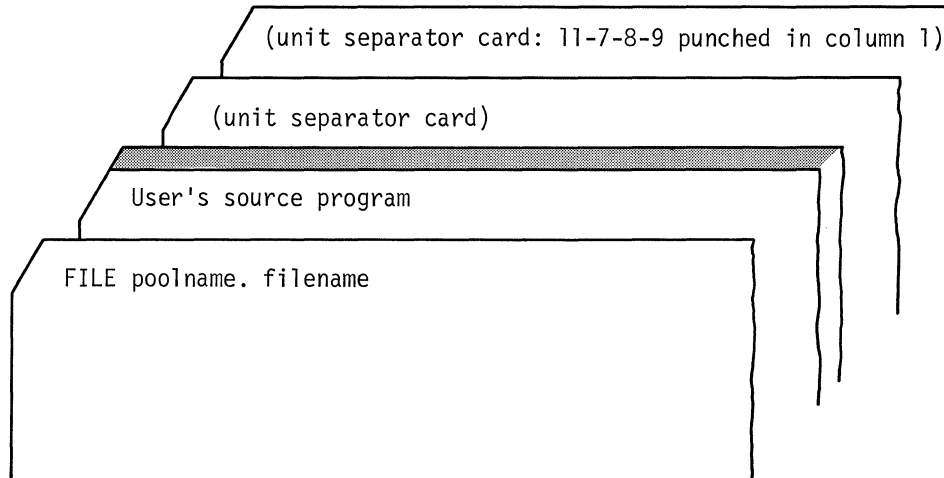
This procedure files the disc assembler object deck on disc. Upon successful completion, the console prints:

A) ENTER PROGRAM NAME

USING THE ASSEMBLER

Filing Source Deck

To prepare for assembly, file the source deck on the DMF disc pack using the following deck setup.



Then, proceed as follows, interacting with the console on either the Workstation or Communications Terminal:

Operator: Place input deck in card reader. Get load condition on console. Depress ENTER key (on Workstation) or control / key (on Communications Terminal).

Console: Prints: A) ENTER PROGRAM NAME

Operator: Type: FILE or REFILE
(Use FILE if poolname.filename on first input card identifies a new file. Use REFILE if the identified file already exists.)

Depress: ENTER key (on Workstation) or Control / key (on Communications Terminal).

This procedure files the source deck on disc. Upon successful completion, the console again prints:

A) ENTER PROGRAM NAME

Assembling Source Program

To assemble the user source program, call the disc assembler into memory and give the name of the file containing the source program, as follows:

Operator: Get load condition on console.

Depress: ENTER key (on Workstation) or control / key (on Communications Terminal).

Console: Prints: A) ENTER PROGRAM NAME

Operator: Type: ASMBLR

(If the assembler is not filed under SYSPOL.ASMBLR, give poolname.filename used to file it.)

Depress: ENTER key (on Workstation) or control / key (on Communications Terminal).

The assembler now enters memory from disc and starts executing. Continue as follows:

Console: Lights: Enter Light (on Workstation).

Prints: E (on Communications Terminal).

Operator: Elect to use input/output default options specified as fixed assembler parameters, or override these with new parameters.

To use the default parameters, simply depress any control key (on the Workstation) or control p control / (on the Communications Terminal) and the assembly will begin. The default parameters normally result in: reading of the source input from the disc file SOURCE.TEMP, writing of common text output in the object code, ignoring of Debug statements, suppression of the symbol cross-reference listing, output of blanks for the date on the program listing, and output of the object program through the card punch.

USING THE ASSEMBLER

To override any of these default options, enter any of the following assembler parameter statements from the assembler command device.

| <u>Statement</u> | <u>Purpose</u> |
|--------------------------|--|
| DEBUG=YES | To assemble Debug statements. |
| DEBUG=NO | To ignore Debug statements (the normal default parameter). |
| COMTXT=YES | To permit generation of object code in Common Core (the normal default parameter). |
| COMTXT=NO | To suppress generation of object code in Common Core. |
| SOURCE=poolname.filename | To specify a disc file as source input. (The default option is the disc file SOURCE.TEMP; if poolname is not specified, SOURCE will be assumed.) |
| OBJECT=poolname.filename | To direct object output to a disc file. (The default option is the card punch; if poolname is not specified, SOURCE will be assumed.) |
| XREF=YES | To request output of a symbol cross-reference listing. |
| XREF=NO | To suppress output of a symbol cross-reference listing (the normal default option). |
| DATE=mm/dd/yy | To print a date in listing headings. (The normal default option is blanks.) |

After each statement, depress the ENTER key (on the Workstation) or control / key (on the Communications Terminal). If the OBJECT=poolname.filename statement is used to direct object output onto disc, the disc file specified must initially be a null file. The disc pack used must contain the necessary DMF routines (the low core, locator, and disc loader programs). (For further directions, see the Disc Management Facility User's Reference Manual.) If a mistake occurs during entry of any of the statements, depress the ENTER key (on the workstation) or the control / key (on the communications terminal) and re-enter the statement.

Operator: To commence assembly, depress any CTL key (on the Workstation) or control p control / (on the Communications Terminal). The assembler will process the source program designated by the SOURCE=poolname.filename statement. If this statement is omitted, the default file SOURCE.TEMP is used.

APPENDICES

- Appendix A ASSEMBLER DECK PARAMETERS
- Appendix B MACHINE INSTRUCTION STATEMENTS
- Appendix C ASSEMBLER COMMANDS
- Appendix D CONDITION CODES
- Appendix E I/O DEVICE CONTROL CHARACTERS
- Appendix F I/O DEVICE CONDITION CODES
- Appendix G STANDARD DEVICE NUMBERS
- Appendix H DISC ADDRESS
- Appendix I CONVERSION TABLES
- Appendix J BOOTSTRAP CARD AND FOUR-CARD LOADER

APPENDIX A: ASSEMBLER DECK PARAMETERS

ASSEMBLER I (CARD VERSION)

| Parameter Card Column | Meaning | Normal Entry | Effect on Assembly | Equivalent Assembler Parameter Statement |
|-----------------------|--|-----------------------|---|--|
| 8 | Source Device No. | 1 | Takes input from card reader. | |
| 9 | Listing Device No. | 2 | Transmits listings to line printer. | |
| 10 | Object Device No. | 4 | Transmits object program to card punch. | |
| 11 | Not used. | | | |
| 12 - 15 | Lower bound of label table in Common Core. | 0000 | Declares no boundary. | |
| 16 - 19 | Upper bound of label table in Common Core. | 0000 | Declares no boundary. | |
| 20 - 23 | Object card trailer control characters. | SMbb | | |
| 24 - 27 | Maximum number of lines printed on listing page. | 0060 | | |
| 28 - 29 | Not used. | | | |
| 30 - 37 | Default date printed | Blank | Prints no date. | DATE= |
| 38 | Default disposition | \$ | Ignores Debug cards. | DEBUG= NO |
| 39 | Default disposition of Common object text. | Y | Punches Common object text. | COMTXT=YES |
| 40 - 46 | Constants used by the assembler. | (Must not be changed) | | |
| 47 - 70 | Not used. | | | |
| 71 - 76 | Program identifier. | | | |
| 77 - 80 | Card sequence number 0001. | | | |

APPENDIX A: ASSEMBLER DECK PARAMETERS

ASSEMBLER I (DISC VERSION)

| Parameter Card Column | Meaning | Normal Entry | Effect on Assembly | Equivalent Assembler Parameter Statement |
|-----------------------|--|------------------------|--|--|
| 8 - 10 | Not used. | | | |
| 11 | Default Disposition of Symbol Cross-Reference Listing. | N | Suppresses Symbol Cross-Reference Listing. | XREF=NO |
| 12 - 15 | Lower bound of label table in Common Core. | 0000 | Names no boundary. | |
| 16 - 19 | Upper bound of label table in Common Core. | 0000 | Names no boundary. | |
| 20 - 23 | Object card trailer control characters. | SMBb | | |
| 24 - 27 | Maximum number of lines printed on each listing page. | 0060 | | |
| 28 - 29 | Not used. | | | |
| 30 - 37 | Default date printed on listing. | Blank | Prints no date. | DATE= |
| 38 | Default disposition of debug cards. | \$ | Ignores Debug cards. | DEBUG=NO |
| 39 | Default disposition of Common object text. | Y | Writes Common object text. | COMTXT=YES |
| 40 - 46 | Constants used by the assembler. | (Must not be changed). | | |
| 47 - 70 | Not used. | | | |
| 71 - 76 | Program identifier. | | | |
| 77 - 80 | Card sequence number 0001. | | | |

APPENDIX B: MACHINE INSTRUCTION STATEMENTS

| INSTRUCTION NAME | OP CODE | OPERAND FORM | FORM NAME |
|------------------|---------|-------------------|---------------------|
| Add | 0100 A | | |
| Divide | 0101 D | | |
| Form Numeric | 1101 FN | A(LA,IA),B(LB,IB) | Two-length form |
| Multiply | 0110 M | | |
| Subtract | 0111 S | | |
| Compare | 1110 C | | |
| Exchange | 1111 X | | |
| Edit | 1100 E | A(L,IA),B(,IB) | One-length form |
| Move Character | 1000 MC | | |
| Move Numeric | 1001 MN | | |
| Read | 0000 R | A(D,IA),B(C,IB) | Device/Channel form |
| Write | 0001 W | | |
| <u>BRANCH</u> | | | |
| Conditional | 1011 BC | A(V),B(V) | |
| Link | 1011 BC | A(6),B(V) | |
| Service Request | 1011 BC | A(7),B(O) | Branch form |
| Unconditional | 1011 BC | A(5) | |
| Switch | 1011 BC | A(8) | |

APPENDIX B: MACHINE INSTRUCTION STATEMENTS

LEGEND (for Machine Instructions)

| | |
|-------|---|
| A,B | Address-A, Address/Count-B |
| | permissible forms: |
| | symbol |
| | symbol constant |
| | (flagged) constant |
| | (flagged) constant constant |
| | * |
| | * constant |
| C | Channel and Control (range 0-5) |
| | (0-FAC, 1-IOC, 2-FAC-control, 3-IOC-control, 5-IOC-non-fill) |
| D | Input/Output Device (range 0-9) |
| IA,IB | Index register (range 0-3, 0 = no indexing) |
| L | Length (range 1-100) |
| LA,LB | Length (range 1-10) |
| V | Branch variant |
| | 0---no branch |
| | 1---branch if condition code is 1 |
| | 2---branch if condition code is 2 |
| | 3---branch if condition code is 3 |
| | 4---branch if condition code is 4 |
| | 5---branch unconditionally |
| | 8---branch and switch unconditionally |
| | 9---no branch. |

INTERNAL INSTRUCTION FORMAT

| | | | | | | | | | | | |
|-------------|----|----|----|----|-----|-----|-----|-----|----|---|----------|
| 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 | 1 | 5 |
| | | | | | | | | | | | 4 |
| LA | A3 | A2 | A1 | A0 | LB | B3 | B2 | B1 | B0 | | 3 |
| | | | | | | | | | | | 2 |
| | | | | | | | | | | | 1 |

APPENDIX C: ASSEMBLER COMMANDS

```
COMMON
END
EJECT
EXEC   xxxx
NORMAL
label  ORG   xxxx
SPACE  xx
TITLE 'text---up to 60 characters'
```

Define Memory (assembler instruction)

```
label  DM   RTS'DATA'(L,I)
```

LEGEND

| | | |
|------|--|----------|
| R | Repetition factor (range: 0-9999) | Optional |
| T | Data Type | REQUIRED |
| | C: character | |
| | N: numeric | |
| | A: address | |
| S | Size of DATA (range: 1-9999) | Optional |
| DATA | Initialization data | Optional |
| L | Length specification for label table (range: 1-100) | Optional |
| I | Index register specification for label table (range: 0-3) | Optional |

RULE

R times S may not exceed 9999.

APPENDIX D: CONDITION CODES

| Instruction | CC=1 | CC=2 | CC=3 | CC=4 |
|---|-----------|--------------|--------------|------------|
| ADD | Minus | Zero | Plus | Overflow |
| BRANCH CONDITIONAL | -- | -- | -- | -- |
| COMPARE | A is less | Equal | A is greater | A not less |
| DIVIDE | Minus | Zero | Plus | Overflow |
| EDIT | Minus | Zero | Plus | -- |
| EXCHANGE | -- | 2 always set | -- | -- |
| FORM NUMERIC | Minus | Zero | Plus | Overflow |
| MOVE CHARACTER | -- | 2 always set | -- | -- |
| MOVE NUMERIC | -- | 2 always set | -- | -- |
| MULTIPLY | Minus | Zero | Plus | -- |
| READ | Error | Normal | Flag | Fault |
| SUBTRACT | Minus | Zero | Plus | Overflow |
| WRITE | Error | Normal | Flag | Fault |
| 1,2,3, 1&4, 2&4, 3&4, (possible values) | | | | |

APPENDIX E: I/O DEVICE CONTROL CHARACTERS

| Device | Character | Definition |
|---------------|----------------|---------------------------------|
| Card Reader | ? | Local Query |
| Card Punch | _ (underscore) | Punch Unit Separator |
| | L or J | Feed a Card |
| | ? | Local Query |
| Line Printer | /nn | Set Vertical Tab Register |
| | K | Advance Paper |
| | L or R | Skip to Top-of-Form |
| | ? | Local Query |
| Magnetic Tape | T | Write Tape Mark |
| | R | Rewind |
| | B | Backspace Block |
| | F | Forwardspace Block |
| | E | Erase |
| | M | Change Mode |
| | L | Switch to Local (and Rewind) |
| | D | "Read Only" Query |

APPENDIX F: I/O DEVICE CONDITION CODES

| | |
|-----------------|---------|
| Possible | 1 |
| Condition Codes | 2 |
| | 3 |
| | 4 and 1 |

Card Reader

| | |
|------------|---|
| 2 (NORMAL) | OK. |
| 3 (FLAG) | Unit Separator character read. |
| 4 (FAULT) | ? sent by a Write Control instruction when the card reader was in Local, card reader addressed by a Write instruction, card reader's power off, or card reader's buffer is empty. |

Card Punch

| | |
|------------|---|
| 2 (NORMAL) | OK. |
| 4 (FAULT) | ? sent by a Write Control instruction when the card punch was in Local, or card punch addressed by a Read instruction, or card punch's power off. |

Line Printer

| | |
|------------|---|
| 2 (NORMAL) | OK. |
| 3 (FLAG) | Write operation or skipping operation completed just as the bottom of the page was sensed. |
| 4 (FAULT) | ? sent by a Write Control instruction when the line printer was in Local, or line printer addressed by a Read instruction, or line printer's power off. |

APPENDIX F: I/O DEVICE CONDITION CODES

Disc

- | | | |
|---|----------|--|
| 1 | (ERROR) | Error during reading, or seek to an invalid track. |
| 2 | (NORMAL) | OK. |
| 3 | (FLAG) | Error during writing. |
| | | OR |
| | | Tried to read a "bad" sector. |
| 4 | (FAULT) | Disc drive not ready. |

Magnetic Tape Read Operation

- | | | |
|---|----------|--|
| 1 | (ERROR) | Parity error(s) detected. |
| 2 | (NORMAL) | OK. |
| 3 | (FLAG) | Tape Mark read and no parity errors were detected. |
| | | OR |
| | | Oversized record encountered and no parity errors were detected. |
| 4 | (FAULT) | Tape drive was switched to Local during the Read operation. |
| | | OR |
| | | Another tape drive with the same address was switched On-Line during the Read operation. |
| | | OR |
| | | Tape drive not ready. |

Magnetic Tape Write Operation

- 1 (ERROR) Parity error(s) detected.
- 2 (NORMAL) OK.
- 3 (FLAG) End-of-Tape reflective marker sensed and no parity errors were detected.
- 4 (FAULT) Attempted to write on a "Read Only" tape.

OR

The tape drive was switched to Local during the Write operation.

OR

Another tape drive with the same address was switched On-Line during the Write operation.

OR

Tape drive not ready.

Magnetic Tape Control Operation

- 2 (NORMAL) OK.
- 3 (FLAG) Tape Mark was spaced over during either Forwardspace Block or Backspace Block.

OR

D sent by a Write Control Instruction when a "Read Only" tape was mounted.

- 4 (FAULT) The tape drive was switched to Local during the control operation.

OR

The tape was already positioned at a load point when backspacing was requested.

OR

Tape drive not ready.

APPENDIX G: STANDARD DEVICE NUMBERS

| | IOC | FAC |
|---|-------------------|--------------------------------|
| 0 | Workstation | Disc Controller |
| 1 | Card Reader | Magnetic Tape Drive 1 |
| 2 | Line Printer | Magnetic Tape Drive 2 |
| 3 | Paper Tape Reader | Magnetic Tape Drive 3 |
| 4 | Card Punch | Magnetic Tape Drive 4 |
| 5 | - | - |
| 6 | Paper Tape Punch | - |
| 7 | Workstation | - |
| 8 | - | On Line Communications Adapter |
| 9 | - | - |

APPENDIX H: DISC ADDRESS

Bit-7 of each character may be either ON or OFF.

Bit-5 of each character must always be ON.

Sectors 00-49 are accessed by the upper of the two heads on each arm and sectors 50-99 by the lower.

The lowest of the five arms is number 4 and the uppermost is 0.

The outermost track is number 000 and the innermost is 199.

| Character | 1 | 2 | 3 | 4 | 5 | 6 | Bit |
|-----------|---|---|---|---|---|---|-----|
| | A | B | D | E | F | G | 4 |
| | A | B | D | E | F | G | 3 |
| | A | B | D | E | F | G | 2 |
| | A | C | D | E | F | G | 1 |

A: Device number (0-9).

B: Arm number (0-4).

C: Hundreds Digit (0 or 1) of a three-digit track number.

D: Tens digit (0-9) of a three-digit track number.

E: Units digit (0-9) of a three-digit track number.

F: Tens digit (0-9) of a two-digit sector number.

G: Units digit (0-9) of a two-digit sector number.

APPENDIX I: CONVERSION TABLES

| SYSTEM TEN INTERNAL CODE | CHAR- ACTER | NAME OF CHARACTER | HOLLERITH CODE | SPECIAL INTERPRETATION |
|-----------------------------|----------------|-----------------------|-------------------|--|
| b b b b b b 7 5 4 3 2 1 | | | | |
| 0 0 0 0 0 0 | SP | Space | No punches | |
| 0 0 0 0 0 1 | ! | Exclamation point | 12-8-7 | Appears as l (logical or on IBM 029 Key punch board. |
| 0 0 0 0 1 0 | " | Quotation marks | 8-7 | |
| 0 0 0 0 1 1 | # | Number sign | 8-3 | Prints as £ on some Model 50 Line Printers. |
| 0 0 0 1 0 0 | \$ | Dollar sign | 11-8-3 | |
| 0 0 0 1 0 1 | % | Percent | 0-8-4 | |
| 0 0 0 1 1 0 | & | Ampersand | 12 | |
| 0 0 0 1 1 1 | ' | Prime, Apostrophe | 8-5 | |
| 0 0 1 0 0 0 | (| Left parenthesis | 12-8-5 | |
| 0 0 1 0 0 1 |) | Right parenthesis | 11-8-5 | |
| 0 0 1 0 1 0 | * | Asterisk | 14-8-4 | |
| 0 0 1 0 1 1 | + | Plus sign | 12-8-6 | |
| 0 0 1 1 0 0 | , | Comma | 0-8-3 | |
| 0 0 1 1 0 1 | - | Hyphen, Minus sign | 11 | |
| 0 0 1 1 1 0 | . | Period, Decimal point | 12-8-3 | |
| 0 0 1 1 1 1 | / | Slash | 0-1 | |
| 0 1 0 0 0 0 | 0 | Zero | 0 | |
| 0 1 0 0 0 1 | 1 | One | 1 | |
| 0 1 0 0 1 0 | 2 | Two | 2 | |
| 0 1 0 0 1 1 | 3 | Three | 3 | |
| 0 1 0 1 0 0 | 4 | Four | 4 | |
| 0 1 0 1 0 1 | 5 | Five | 5 | |
| 0 1 0 1 1 0 | 6 | Six | 6 | |

APPENDIX I: CONVERSION TABLES

| SYSTEM TEN INTERNAL CODE | CHAR- ACTER | NAME OF CHARACTER | HOLLERITH CODE | SPECIAL INTERPRETATION |
|-----------------------------|----------------|----------------------|-------------------|--|
| b b b b b b 7 5 4 3 2 1 | | | | |
| 0 1 0 1 1 1 | 7 | Seven | 7 | |
| 0 1 1 0 0 0 | 8 | Eight | 8 | |
| 0 1 1 0 0 1 | 9 | Nine | 9 | |
| 0 1 1 0 1 0 | : | Colon | 8-2 | |
| 0 1 1 0 1 1 | ; | Semicolon | 11-8-6 | |
| 0 1 1 1 0 0 | < | Less-than sign | 12-8-4 | |
| 0 1 1 1 0 1 | = | Equal sign | 8-6 | Prints as # on some Model 50 line printers. |
| 0 1 1 1 1 0 | > | Greater-than sign | 0-8-6 | |
| 0 1 1 1 1 1 | ? | Question mark | 0-8-7 | |
| 1 0 0 0 0 0 | @ | At sign | 8-4 | |
| 1 0 0 0 0 1 | A | | 12-1 | |
| 1 0 0 0 1 0 | B | | 12-2 | |
| 1 0 0 0 1 1 | C | | 12-3 | |
| 1 0 0 1 0 0 | D | | 12-4 | |
| 1 0 0 1 0 1 | E | | 12-5 | |
| 1 0 0 1 1 0 | F | | 12-6 | |
| 1 0 0 1 1 1 | G | | 12-7 | |
| 1 0 1 0 0 0 | H | | 12-8 | |
| 1 0 1 0 0 1 | I | | 12-9 | |
| 1 0 1 0 1 0 | J | | 11-1 | |
| 1 0 1 0 1 1 | K | | 11-2 | |
| 1 0 1 1 0 0 | L | | 11-3 | |
| 1 0 1 1 0 1 | M | | 11-4 | |
| 1 0 1 1 1 0 | N | | 11-5 | |
| 1 0 1 1 1 1 | O | | 11-6 | |
| 1 1 0 0 0 0 | P | | 11-7 | Can also be used internally as -0. |
| 1 1 0 0 0 1 | Q | | 11-8 | Can also be used internally as -1. |
| 1 1 0 0 1 0 | R | | 11-9 | Can also be used internally as -2. |

APPENDIX I: CONVERSION TABLES

| SYSTEM TEN INTERNAL CODE | CHAR- ACTER | NAME OF CHARACTER | HOLLERITH CODE | SPECIAL INTERPRETATION |
|-----------------------------|----------------|----------------------|-------------------|---|
| b b b b b b 7 5 4 3 2 1 | | | | |
| 1 1 0 0 1 1 | S | | 0-2 | Can also be used internally as -3. |
| 1 1 0 1 0 0 | T | | 0-3 | Can also be used internally as -4. |
| 1 1 0 1 0 1 | U | | 0-4 | Can also be used internally as -5. |
| 1 1 0 1 1 0 | V | | 0-5 | Can also be used internally as -6. |
| 1 1 0 1 1 1 | W | | 0-6 | Can also be used internally as -7. |
| 1 1 1 0 0 0 | X | | 0-7 | Can also be used internally as -8. |
| 1 1 1 0 0 1 | Y | | 0-8 | Can also be used internally as -9. |
| 1 1 1 0 1 0 | Z | | 0-9 | |
| 1 1 1 0 1 1 | [| Opening bracket | 12-8-2 | Appears as ¢ on IBM 029 Keypunch board. |
| 1 1 1 1 0 0 | \ | Reverse slash | 0-8-2 | Appears as 0-2-8 key on IBM 029 Keypunch board. |
| 1 1 1 1 0 1 |] | Closing bracket | 11-8-2 | Appears as ! on IBM 029 Keypunch board. |
| 1 1 1 1 1 0 | ^ | Circumflex | 11-8-7 | Appears as ¬ (logical not) on IBM 029 Keypunch board. Prints as † on some Model 50 Line Printers. |
| 1 1 1 1 1 1 | _ | Underline | 0-8-5 | Prints as ← on some Model 50 Line Printers. |

APPENDIX J: BOOTSTRAP CARD AND FOUR-CARD LOADER

Bootstrap Card and 4 Card Loader

```
0001010290  BOOTSTRAP CARD **USE IF CARD READER IS DEVICE 0**
:POOP010070POPP050130: 0000      0000 WP1X940182:1P18840182:0010110080:ROQY050070:
:001W710009 *      1P08540031:1P12040045:PPP0TQP189:ROQU050110:P003010010:R001170012
:POPP050050* 0000TS10:PPW7740182:RORQ050090:PPQ0110186:RORS050260:P010350255:P01P210250
:PO10810000:PPW0110187:RORX050060:P010240296:POPP050000:00000000
1          10          20          30          40          50          60          70          80
```


GLOSSARY

GLOSSARY

INTRODUCTORY NOTE

The purpose of the Glossary is to define all new terms introduced in the text and to define any special use made of standard terms. Standard terms which are used in a standard sense are not included. For elucidation on these, the reader is referred to Computer Dictionary and Handbook by Charles J. Sippl (Howard W. Sams & Co., Inc., Indianapolis, 1966).

Following is a list of the items defined in the Glossary:

| | |
|--------------------------------------|---------------------------|
| Alphabetic Field | Host Partition |
| Arithmetic and Control Unit (ACU) | Implied Parameter |
| Auxiliary Storage | Index Register |
| Block | IOC |
| Bootstrapping | Label Table |
| Branch | Link |
| Buffer | Literal |
| Burst Mode Transmission | Local Mode |
| Byte | Main Memory |
| Byte Mode Transmission | Memory Module |
| Central Processing Unit (CPU) | Mixed Field |
| Channel | Multiprogramming |
| Characters | Numeric Field |
| Common Area of Memory | Object Program |
| Control Character | On-Line Mode |
| Control Field | Operation Code |
| Cycle-Stealing | Overdraft |
| Disc, Bound | Overflow |
| Disc, Free | Parity Bit |
| Double Frame | Partition |
| Effective Address | Partition Switching |
| File | Privileged Area of Memory |
| Filler Characters | Protected Area of Memory |
| Flagged Constant | Return Address |
| Flowcharting Symbols | Sector |
| Hexadecimal Number System | Source Program |
| | Symbol |
| | USASCII |

A

Alphabetic Field

A field consisting strictly of the alphabetic characters A thru Z.

Arithmetic and Control Unit (ACU)

In System Ten, that part of the Central Processing Unit (CPU) which controls and performs the execution of machine instructions.

Auxiliary Storage

Storage in addition to the main storage of a computer. Auxiliary storage usually holds much more information than the main storage, and the information is accessible less rapidly. In System Ten, the disc is considered to be auxiliary storage.

B

Block

When used in connection with the Friden Model 45 Magnetic Tape Drive, block means a series of consecutive tape characters. The end of one block and the beginning of the next is signalled by a segment of blank tape called an inter-block gap. By this definition, block is synonymous with the phrase "physical record."

Bootstrapping

A technique for loading the first few instructions of a program into storage, then using these instructions to bring the rest of the program into storage. This sometimes involves either the manual entering of a few instructions or the use of a key on a console. In System Ten, the bootstrap sequence is initiated by the depression of a LOAD button on an input device or by the occurrence of certain errors during program execution.

Branch

In System Ten, a departure from the normal sequential processing of instructions as caused by the execution of the Branch instruction. Another type of departure from sequential processing is the switch which passes control to a neighboring partition.

Buffer

Temporary storage used to compensate for the difference in operating speeds of input/output devices and the Central Processing Unit (CPU). In System Ten, the card reader has two card buffers, the card punch has three card buffers, the line printer has two line buffers, etc. Each IOC has one character buffer.

Burst Mode Transmission

A mode of communication between the Central Processing Unit (CPU) and external input/output devices. The information is transmitted without interruption as a solid procession of binary bits. In System Ten, the burst mode is employed in transmitting between Main Memory and the Friden Model 40 Disc Drive.

Byte

In System Ten, a group of 6 adjacent binary bits. The bits are referred to as bit-7, bit-5, bit-4, bit-3, bit-2, and bit-1. Bit-6 of the USASCII Standard Code is not used.

Byte Mode Transmission

A mode of communication between the Central Processing Unit (CPU) and external input/output devices. Transmission proceeds one character at a time on a cycle-stealing basis. In System Ten, all transmission through the Input/Output Channel (IOC) is accomplished in this mode, as is also transmission through the File Access Channel (FAC) when the Friden Model 45 Magnetic Tape Drive is the input/output device.

C**Central Processing Unit (CPU)**

In System Ten, the Central Processing Unit (CPU) comprises the Arithmetic and Control Unit (ACU), the File Access Channel (FAC), one to twenty Input/Output Channels (IOC), and Main Memory.

Channel

A path along which information, particularly a series of bits or characters, may flow. In System Ten, each partition has a private Input/Output Channel (IOC) which transmits in the byte mode. Common to all partitions is a single File Access Channel (FAC) which transmits in the burst mode when the disc is used.

Characters

A set of coded symbols that includes the decimal digits 0 thru 9, letters A thru Z, punctuation marks, operation symbols, and other symbols. In System Ten, each character is represented by 6 binary bits.

GLOSSARY

Common Area of Memory

In System Ten, that portion of memory which is not partitioned. The Common Area comprises Protected Storage (locations 0-299), a Non-Privileged area, and an optional Privileged area accessible only to privileged partitions.

Control Character

A character whose occurrence in a particular context initiates, modifies, or stops a control operation.

Control Field

In System Ten, the second operand used by the Edit instruction. The control field governs the format of the edited result.

Cycle-Stealing

Data channels give the Arithmetic Control Unit (ACU) the ability to delay the execution of a program for communication of an input/output device with memory. If an input unit requires a memory cycle to store data that it has collected, the data channel makes it possible to delay the program prior to the execution of an instruction, and to store the data without changing the logical condition of the ACU. After the data is stored, the program continues as though nothing has occurred. In System Ten, cycle-stealing occurs between instructions, and during the transmission of data between magnetic tape and memory.

D

Disc, Bound

In System Ten, a disc is bound to a given partition as soon as the partition institutes a seek upon it. The disc remains bound until data transmission is complete. While a disc is bound to a given partition, it cannot be accessed by another partition.

Disc, Free

In System Ten, a disc is free when it is not bound to another partition.

Double Frame

In System Ten, a special method of reading or writing 9-track magnetic tape in which the eight bits of data in each tape row are constructed from (or read into) the numeric portions of two consecutive locations in main memory. The same method can also be used on eight-channel paper tape readers and paper tape punches.

E

Effective Address

The address that is actually used in a particular execution of an instruction. In System Ten the effective address for a given instruction address field is the sum of the address field added to the contents of whatever index register is associated with that field.

F

File

When used in connection with the Friden Model 45 Magnetic Tape Drive, file means a series of consecutive blocks (or "physical records"). The end of one file and the beginning of the next is signalled by a special two character block called a Tape Mark.






Filler Characters

All characters used in the edit machine instruction EXCEPT the 'at' sign (@), slash (/), dash (-), comma (,), and decimal point (.).

Flagged Constant

In System Ten Assembler-1 Language, a constant with a C or P appended to indicate a common or partition address. E.G., 495C, 4567P.

Flowcharting Symbols

| SYMBOL | REPRESENTS |
|---|---|
|  | PROCESSING A group of program instructions which perform a processing function of the program. |
|  | DECISION The decision used to document points in the program where a branch to alternate paths is possible based upon variable conditions. |
|  | TERMINAL The beginning, end, or point of interruption in a program. |
|  | CONNECTOR An entry from, or an exit to, another part of the program flowchart. |
|  | FLOW DIRECTION The direction of processing or data flow. |

H

Hexadecimal Number System

A number system using the equivalent of the decimal number sixteen as a base. In System Ten, the digits greater than 9 are written as 10, 11, 12, 13, 14, 15.

Host Partition

The partition in control when the given instruction is executed.

I

Implied Parameters

The length and index register parameters associated with a label in the label table. Whenever a label appears in the operand field of a statement using length and/or index parameters, the length and index register parameters may be omitted and they will be supplied automatically from the label table. Thus the idea that, by being omitted, the parameters were "implied."

Index Register

A register whose primary purpose is to modify addresses in computer instructions. In System Ten there are three index registers in each partition. They are referred to by number and occupy partition storage locations 11-14 (index register 1), 21-24 (index register 2), and 31-34 index register 3).

IOC

An abbreviation for Input/Output Channel on the System Ten.

L

Label Table

A table produced by the assembler which contains, for every label in the source program, the address assigned to the label by the assembler, an indication as to whether the address is in common or partition, an implied length associated with the label, and the implied index.

Link

In System Ten, a variant of the Branch instruction ordinarily used in calling subroutines. Link first establishes a return path from a subroutine and then passes control to the beginning of the subroutine.

Literal

An item of data with its value as stated. The assembler permits the use of a literal in the text field of the TITLE assembler command and in the data field of the DM (Define Memory) assembler command. Literals are not used in the machine instruction statements.

Local Mode

The mode in which data may NOT be transmitted between a device and an Input/Output Channel (IOC).

M**Main Memory**

In System Ten, the entire core storage. This includes the storage occupied by all partitions as well as the entire common region which includes the protected area, the non-privileged area, and the privileged area.

Memory Module

In System Ten, the minimum unit of core storage. A Memory Module contains 10,000 character positions. In System Ten, core storage may contain 1-11 Memory Modules.

Mixed Field

A field which may include any combination of characters including, for example, alphabetic, numeric, and punctuation characters. In System Ten, a field prepared for printing by the Edit instruction is usually a mixed field.

Multiprogramming

A technique for handling numerous routines or programs simultaneously by overlapping or interleaving their execution. In System Ten, the programs being executed simultaneously each reside in a separate memory partition. Multiprogramming is achieved by passing control from one partition to another in round-robin sequence.

N**Numeric Field**

A field containing numeric information and sign indication.

O

Object Program

The machine-language representation of a source program. If a source program contains no errors, detectable by the assembler, it is translated to machine-language; the object program is what results from this translation process.

On-Line Mode

The mode which permits the transmission of data between a device and an Input/Output Channel (IOC) or File Access Channel (FAC).

Operation Code

The part of a System Ten machine instruction which specifies the operation to be performed.

Overdraft

In System Ten, an intermediate condition which sometimes occurs in the subtraction process. Not to be confused with Overflow, which yields a wrong answer.

Overflow

In System Ten, an erroneous result caused by an attempt to develop an answer too large for the field assigned to it.

P

Parity Bit

A binary digit (i.e., either 0 or 1) appended to a string of bits to make the sum of all the bits which are ON either always odd or always even.

Partition

In System Ten, a portion of core storage. A system may contain 1-20 partitions. Each partition has 3 index registers and an Input/Output Channel (IOC). Partitions may communicate with each other only through common storage or devices on the File Access Channel (FAC).

Partition Switching

In System Ten, an automatic process by which control passes from one partition to its neighbor. Partition Switching consists essentially of saving status information necessary to resume the program which is relinquishing control, selecting the partition which is to gain control, restoring its Condition Code, and passing control to the appropriate instruction within it.

Privileged Area of Memory

In System Ten, an optional hardware setting that reserves an upper portion of common storage for use by privileged partitions which are designated when the option is set.

Protected Area of Memory

In System Ten, locations 0-299 of the common storage area. Programs cannot store information in this area which is used by the ACU to keep information pertinent to partition switching and input/output operations. A program may examine information in the protected area even though it cannot (directly) alter it.

R**Return Address**

In System Ten, the address of the instruction to which control returns after a particular execution of a subroutine.

S**Sector**

A sector is one-fiftieth (1/50th) of a track on the Friden Model 40 Disc Drive. Each sector holds 100 characters (each consisting of 6 bits). Reading and writing on the disc is always done in groups of 100 characters.

Source Program

A program coded in other than machine-language, and which must therefore be translated into machine-language by an assembler or compiler before it can be executed. The assembler processes a source program coded in assembler language and translates it into an object deck.

Symbol

A group of from one to six characters. The first character must be one of the alphabetic characters A through Z, or one of the special character [\]^_ or -. The others can be any of these alphabetic or special characters, or the digits 0 through 9.

U**USASCII**

A contraction for "United States of America Standard Code for Information Interchange." This standard defines the graphics and codes for a 128 - character set. Commonly referred to as ASCII.

INDEX



INDEX

A

- Add instruction, 2-2 to 2-3
- Alignment
 - data fields, 3-7, 3-17
 - instruction, 1-19, 3-17
- Assembler commands, 1-29, sec 3, app C
 - COMMON, 3-2 to 3-3
 - DM (Define Memory), 3-4 to 3-10
 - EJECT, 3-11
 - END, 3-12
 - EXEC, 3-13
 - NORMAL, 3-14 to 3-15
 - ORG, 3-16 to 3-17
 - SPACE, 3-18
 - TITLE, 3-19
- Assembler deck parameters, sec 5, app A
- Assembling source programs
 - card version, 5-4 to 5-6
 - disc version, 5-11 to 5-13
- Assembly process, overview, 1-7 to 1-8
- Asterisk as address, 1-12, 3-3, 3-15
- Auxiliary software, 1-5

B

- Bootstrap card and four-card loader, app J
- Branch instruction, 2-4 to 2-9

C

- Channel indication, 2-27
- Character set, 1-12 to 1-13, 2-12
- Coding forms and source cards, 1-17 to 1-18
- Commands, assembler, 1-29 to 1-30, sec 3
- Comments field, 1-16
- Comments statement, 1-17
- COMMON assembler command, 3-2 to 3-3
- Common indication, 1-22 to 1-24
- Common/partition specification, 1-27 to 1-28
- Compare instruction, 2-10 to 2-12
- COMTXT=YES, COMTXT=NO (assembler parameters), 5-4, 5-12, app A
- Condition codes
 - I/O device, app F
 - machine instructions, app D
- Constants and symbols, 1-22 to 1-24, 3-2, 3-14
- Control characters, I/O device, app E
- Control statements, 1-29 to 1-30, sec 3, app C
- Conversion tables, app I
- Cross-reference listing, 4-7

INDEX

D

DATE=mm/dd/yy (assembler parameter), 5-4, 5-12, app A
Debug statement, 1-17
DEBUG=YES, DEBUG=NO (assembler parameters), 5-4, 5-12, app A
Deck structure, 1-9
Device numbers, app G
Diagnostics, 1-19, 4-2
Disc address, app H
Divide instruction, 2-13 to 2-14
DM (Define Memory) assembler command, 3-4 to 3-10

E

Edit instruction, 2-15 to 2-18
EJECT assembler command, 3-11
END assembler command, 3-12
Error messages, 4-8
Error statement listing, 4-1 to 4-4
Exchange instruction, 2-19 to 2-20
EXEC assembler command, 3-13

F

Fields, source statement, 1-14 to 1-16
Flagged constant, 3-14, Glossary-5
Form Numeric instruction, 2-21 to 2-22
Format, statement, 1-12

I

Identification field, 1-16
Implied operand forms, 1-24
Index register specification, 1-28
Instruction alignment, 1-19
Instruction fields, 1-26 to 1-29
I/O device condition codes, app F
I/O device control characters, app E
I/O device numbers, app G
I/O device requirements, 1-3 to 1-5
 card version, 1-3
 disc version, 1-5

L

Label field, 1-14 to 1-15
Label table, 1-9, 4-5, Glossary-6
Link variant of branch instruction, 2-7, Glossary-6
Listing output, 4-1 to 4-7
Literal, Glossary-7
Loader, App J
Loading, 3-13, 4-9

M

Machine instruction statements, 1-6, sec 2, app B
 Add, 2-2 to 2-3
 Branch, 2-4 to 2-9
 Compare, 2-10 to 2-12
 Divide, 2-13 to 2-14
 Edit, 2-15 to 2-18
 Exchange, 2-19 to 2-20
 Form Numeric, 2-21 to 2-22
 Move Character, 2-23
 Multiply, 2-25 to 2-26
 Read, 2-27 to 2-32
 Subtract, 2-33 to 2-34
 Write, 2-35 to 2-40
Memory assignment counters, 3-17
Memory requirements, 1-2
Move Character instruction, 2-23
Move Numeric instruction, 2-24
Multiply instruction, 2-25 to 2-26

N

NORMAL assembler command, 3-14 to 3-15

O

Object deck formats, 4-9
Object listing, 4-1
Operand field, 1-16
Operand forms, 1-20 to 1-25
Operation field, 1-15
ORG assembler command, 3-16 to 3-17
Overlay structure, 1-10 to 1-11
Overview of assembly process, 1-7 to 1-8

P

Parameter statements, 1-6 to 1-7, sec 5
Parameters, assembler deck
 card version, 5-1 to 5-4, A-1
 disc version, 5-7 to 5-8, A-2
Program identification field, 1-16
Program listing, 4-6 to 4-7

R

Read instruction, 2-27 to 2-32
Return address, 2-8

INDEX

S

- S card, 4-9
- Sequence number field, 1-16
- Service request, variant of branch instruction, 2-9
- Software required, 1-5
- Source deck structure, 1-9
- Source statement fields, 1-14 to 1-16
- SPACE assembler command, 3-18
- Standard device numbers, app G
- Start address
 - loading, 3-13
 - subroutine, 2-8
- Statement format, 1-12
- Statement repertoire (machine instructions), 1-6, sec 2
- Subtract instruction, 2-33 to 2-34
- Symbol, Glossary-9
- Symbol cross-reference listing, 4-7
- Syntactical errors, 1-19
- System Ten character set, 2-12

T

- T card, 4-9
- Tens boundary alignment, 1-19, 3-17
- TITLE assembler command, 3-19

U

- USASCII, 1-12, Glossary-9
- USASCII code chart, 1-13
- Using the assembler, sec 5
 - card version, 5-1 to 5-6
 - disc version, 5-7 to 5-13

W

- Write instruction, 2-35 to 2-40

SINGER
FRIDEN DIVISION

Publication No. 40-029-1
(Control No. B004PB)