PAL III SYMBOLIC ASSEMBLER
PDP-8 PROGRAMMING MANUAL

For additional copies order No. DEC-08-ASAC-D from Program Library,
Digital Equipment Corporation, Maynard, Massachusetts Price $1.00

DIGITAL EQUIPMENT CORPORATION • MAYNARD, MASSACHUSETTS

1st Printing August 1965
2nd Printing Rev June 1967
3rd Printing November 1967
4th Printing Rev May 1968
5th Printing Printing October 1968
6th Printing February 1969

Instruction times, operating speeds and the like are included in this manual for reference only; they are not to be taken as specifications.

PREFACE

The PDP-8 comes to the user complete with an extensive selection of system programs and routines making the full data processing capability of the new computer immediately available to each user, eliminating many commonly experienced initial programming delays.

The programs described in these abstracts come from two sources, past programming effort on the PDP-5 computer, and present and continuing programming effort on the PDP-8. Thus the PDP-8 programming system takes advantage of the many man-years of program development and field testing by PDP-5 users.

Although in many cases PDP-8 programs originated as PDP-5 programs, all utility and functional program documentation is issued in a new, recursive format introduced with the PDP-8.

Programs written by users of either the PDP-5 or the PDP-8 and submitted to the users' library (DECUS - Digital Equipment Corporation Users' Society) are immediately available to PDP-8 users.

Consequently, users of either computer can take immediate advantage of the continuing program developments for the other.

CONTENTS

Chapter		Page
1	INTRODUCTION	1-1
2	ILLUSTRATIONS OF PDP-8 ASSEMBLER FEATURES	2-1
	The Location Counter	2-1
	Coding Illustrations	2-1
3	THE SOURCE LANGUAGE	3-1
	The Character Set	3-1
	Letters	3-1
	Digits	3-1
	Punctuation Characters	3-1
	Ignored Characters	3-2
	Illegal Characters	3-2
	Elements	3-2
	Number	3-2
	Symbol	3-3
	Parameter Assignments	3-3
	Symbol Definition	3-4
	Expressions	3-5
	Current Address Indicator	3-8
	Comments	3-9
	Pseudo-Instructions	3-9
4	PROGRAM PREPARATION AND ASSEMBLER OUTPUT	4-1
	Program Tape	4-1
5	OPERATING INSTRUCTIONS	5-1
	Summary	5-2
6	SYMBOL TABLE ALTERATION	6-1

CONTENTS (continued)

Appendix		Page
1	SYMBOL LISTS	A1-1
2	ASCII CHARACTER SET	A2-1

CHAPTER 1

INTRODUCTION

The use of an assembly program has become standard practice in the programming of digital computers. Use of an assembler permits a programmer to code in a more convenient language than basic machine code. The advantages of this practice are widely recognized: Easily recognized mnemonic codes are used instead of numeric codes; instructions or data may be referred to by a symbolic name; decimal data may be used as such with the assembler making the required decimal-to-binary conversion; programs may be altered without extensive changes in the source language; and debugging is simplified.

The basic process performed by the Assembler is the substitution of numeric values for symbols, according to associations found in the symbol table. In addition, the user may request that the Assembler itself assign values to the user's own symbols at assembly time. These symbols are normally used to name memory locations, which may then be referenced by name.

The ability to use mnemonic names to represent machine instructions is of great value. The name TAD reminds the user of the Two's complement ADdition instruction, while the number 1ØØØ does not. Consequently, the instructions are easier to remember when mnemonics are used. The same is true of location names. It is much easier to associate the name TOTAL with the location containing the accumulated total than it is to remember that location 1374 contains the total.

Another advantage is that, since the assignment of absolute numbers to symbolic locations is done by the Assembler, the updating of a program by adding or removing instructions is simplified.

In addition to translating statements directly into their binary equivalents, the Assembler will accept instructions for performing translations. These instructions may not look different from other instructions, but they do not generate binary codes. For this reason, they are referred to as pseudo-instructions. For example, the pseudo-instruction DECIMAL tells the Assembler that all numbers following in the program are to be taken as decimal rather than as octal. This instruction is important to the assembly process but has no binary equivalent in the object program. Certain other features of assembly can be directed to the Assembler by the setting of the switch register, abbreviated SR.

The PDP-8 Assembly System consists of the Assembler (PAL III) and the Binary Loader (DEC-08-LBAA-PM). A source program prepared in the source language using ASCII code is translated by the Assembler into a binary object tape in two passes through the Assembler. The object binary tape is loaded by the Binary Loader into the computer ready for execution.

During the first pass of the assembly, all symbols are defined and placed in the Assembler's symbol table. During the second pass, the binary equivalents of the input source language are generated and punched. The Assembler has an optional third pass, which produces an "assembly listing," or a listing with the location, generated binary, and source code side by side on a line.

The PDP-8 Assembly system also includes the Symbolic Tape Editor (DEC-08-ESAB-D) for altering or editing the source language tape; the DEC Debugging Tape (DDT-8, DEC-08-CDDA-D) for debugging the object program by communicating with it in the source language, and various other utility programs such as dumps, etc.

The Assembler requires a basic PDP-8 system consisting of the ASR33 Tape Reader and Punch and a 4K core memory. The Assembler can use either the High-Speed Reader, the High-Speed Punch, or both. The basic Assembler allows 590 user symbols when using the ASR33 and allows 495 user symbols when using the high speed reader. The Extended Assembler contains additional symbols for all optional devices. This symbol list is to be found in the Appendix.

CHAPTER 2

ILLUSTRATIONS OF PDP-8 ASSEMBLER FEATURES

THE LOCATION COUNTER

In general, statements generate 12-bit binary words which are placed into consecutive memory locations when the object tape is loaded. The location counter is a register used by the PDP-8 Assembler to keep track of the next memory location available. It is updated after processing each statement. The location counter may be explicitly set by an element or expression preceded by an asterisk. The element or expression following the asterisk sets the current location counter to the value of that element or expression. Subsequent instructions are assembled into subsequent locations.

Example:

```
*3ØØ
```

The next instruction would be placed in location 3ØØ. The location counter is initially set to Ø2ØØ.

CODING ILLUSTRATIONS

To illustrate some of the features of the PDP-8 Assembler, a small routine has been chosen and coded in a number of different ways. The routine continually adds 1 to the contents of a location until the result is positive, then halts. The instructions used are represented as their octal codes (more compact than the binary actually used). The number being incremented is in location 17Ø. The notation C(A) means the contents of location A.

```
*1ØØ    117Ø    /C(17Ø) INTO AC
*1Ø1    7ØØ1    /ADD 1 TO AC
*1Ø2    317Ø    /STORE IN LOCATION 17Ø
*1Ø3    117Ø    /FETCH C(17Ø)
*1Ø4    771Ø    /SKIP ON POSITIVE AC, CLEAR AC
*1Ø5    51ØØ    /JUMP TO LOCATION 1ØØ
*1Ø6    74Ø2    /HALT
*17Ø    Ø       /WILL CONTAIN NUMBER TO BE INCREMENTED
```

Since the location counter is automatically incremented, specifying sequential addresses could have been avoided after the first address in the progression. In addition, the names of the PDP-8 instructions could have been used in place of the octal codes. The octal representation of these instructions is substituted by the Assembler whenever symbols appear in the program.

Example 2:

```
*1ØØ
          TAD     17Ø
          IAC
          DCA     17Ø
          TAD     17Ø
          SPA     CLA
          JMP     1ØØ
          HLT
*17Ø      Ø
```

The same program could have been written using symbolic address tags. The comma after the symbol A indicates to the Assembler that the location in which it places the instruction TAD B is to be named A. Information associating the symbol A with the number of actual locations is placed in the Assembler's symbol table. Consequently, when processing the instruction JMP A, the Assembler finds the symbols JMP and A in the symbol table and uses these values to form the binary equivalent of the instruction JMP A.

Example 3:

```
*1ØØ
A,        TAD B
          IAC
          DCA B
          TAD B
          SPA CLA
          JMP A
          HLT
*17Ø
B,        Ø
```

Unless the user specifically wanted to use location 17Ø for storage, he could let the Assembler assign the location.

Example 4:

```
*1ØØ
A,        TAD B
          IAC
          DCA B
          TAD B
          SPA CLA
          JMP A
          HLT
B,        Ø
```

CHAPTER 3

THE SOURCE LANGUAGE

This chapter explains the features of the ASCII source language available to the user of PAL III.

THE CHARACTER SET

Letters

A B C D E...X Y Z

Digits

1 2 3 4 5 6 7 8 9 Ø

Punctuation Characters

Since a number of characters are invisible (i.e. nonprinting), the following notation is used to represent them in the examples:

␣	space
→|	tab
↵	carriage return

The following characters are used to specify operations to be performed upon symbols or numbers:

Character		Use
␣	space	combine symbols or numbers
+	plus	combine symbols or numbers
-	minus	combine symbols or numbers
↵	carriage return	terminate line
→|	tab	combine symbols or numbers or format the source tape
,	comma	assign symbolic address
=	equals	define parameters
*	asterisk	set current location counter
;	semicolon	terminate coding line
$	dollar sign	terminate pass

.	point	has value equal to current location counter
/	slash	indicates start of a comment

Ignored Characters

form feed	end of a logical page of a source program (See Symbolic Editor DEC-08-ESAB-D)
blank tape	used for leader/trailer
rubouts	used for deleting characters
code 2ØØ	used for leader/trailer
line feed	follows carriage return

Illegal Characters

All other characters are illegal and cause the Illegal Character error printout: IC dddd AT dddd during PASS1. The first number is the value of the offending character, and the second is the value of the current location counter where it occurred. Illegal characters are ignored.

ELEMENTS

Any group of letters, digits, and punctuation which represents binary values less than 2^{12} is an element.

Number

Any sequence of numbers delimited by punctuation characters forms a number.

Example:

```
1
12
4372
```

The radix control pseudo-instructions indicate to the Assembler the radix to be used in number interpretation. The pseudo-instruction DECIMAL indicates that all numbers are to be interpreted as decimal until the next occurrence of the pseudo-instruction OCTAL.

The pseudo-instruction OCTAL indicates that all numbers are to be interpreted as octal until the next occurrence of the pseudo-instruction DECIMAL. The radix is initially set to octal and remains octal unless otherwise specified.

Symbol

Any sequence of letters and digits beginning with a letter and delimited by punctuation characters is a symbol. Although a symbol may be any length, only the first six characters are considered, and any additional characters are ignored; symbols which are identical in their first six characters are considered identical. Pseudo Instructions may not be used as symbols or tags within a program.

The Assembler has in its permanent symbol table definitions of the symbols for all PDP-8 operation codes, operate commands, and many IOT commands (see the Appendix for a complete list). These may be used without prior definition by the user.

Examples:

JMS	is a symbol whose value of 4ØØØ is taken from the operation code definitions.
A	is a user-created symbol. When used as a symbolic address tag, its value is the address of the instruction it tags. This value is assigned by the Assembler.

PARAMETER ASSIGNMENTS

A parameter may be assigned by use of the equal sign. The symbol to the left of the equal sign is assigned the value of the expression on the right.

Examples:

```
A=6
EXIT=RETURN=JMP I Ø
```

Symbols defined by use of the equal sign may be used in any valid expression.

Example:

```
    A=1ØØ
    B=4ØØ
    A+B         has the value 5ØØ
TAD A           has the value 11ØØ
```

If the expression to the left of the equal sign has already been defined, the ReDefinition diagnostic:

```
RD   XXXXXX   AT   dddd
```

Will be typed where XXXXXX is the symbol's name and dddd is the contents of the current location counter at the point of redefinition. The new value will be stored in the symbol table.

Example:

```
*1ØØ
CLA=76ØØ
```

will cause the diagnostic:

```
RD  CLA    AT    Ø1ØØ
```

Whenever CLA is used after this point, it will have the value 76ØØ.

SYMBOL DEFINITION

A symbol may be defined by the user in one of two ways

(1) by use of parameter assignment

Example:

```
DISMIS=JMP I Ø
```

and (2) by use of the comma

When a symbol is terminated by a comma, it is assigned a value equal to the current location counter. If it is defined more than once in this manner, the Assembler will type the duplicate tag diagnostic:

```
DT  XXXXXX  AT  dddd
```

where XXXXXX is the symbol, and dddd is the current location counter at the second occurrence of the attempted symbol definition. The symbol is not redefined.

Example:

```
    *3ØØ
   START,     TAD A
              DCA COUNTER
  CONTIN,     JMS LEAVE
              JMP START
       A,     -74
 COUNTER,     Ø
   START,     CLA CLL
                .
                .
                .
```

The symbol "START" would have a value of Ø3ØØ, the symbol "CONTIN" would have a value of Ø3Ø2, the symbol "A" would have a value of Ø3Ø4, the symbol "COUNTER" (considered by the Assembler to be COUNTE) would have a value of Ø3Ø5, and when the Assembler processed the next line, it would type during PASS1:

```
        DT     START     AT     Ø3Ø6
```

Since the first PASS of PAL III is used to define all symbols in the symbol table, the Assembler will type a diagnostic if, at the end of PASS1, there are any symbols remaining undefined. For example:

```
        *717Ø
A,      TAD C
        CLA CMA
        HLT
        JMP A1
C,      Ø
$
```

would produce the Undefined Address diagnostic:

```
        UA     XXXXXX     AT     dddd
```

where XXXXXX is the symbol and dddd is the location at which it was first seen. The entire symbol table is printed at the end of PASS1. In the case of the above example, this would be:

```
A       717Ø
UA      A1     AT     7173
C       7174
```

If, during PASS1, PAL III detects that its symbol table is full (in other words, that there is no more memory space to store symbols and their associated values), the Symbol Table full diagnostic:

```
        ST     XXXXXX     AT     dddd
```

is typed. XXXXXX is the symbol that caused overflow, and dddd is the current location when the overflow occurred. The Assembler halts and may not be restarted. The source program should be segmented, or more address arithmetic used, to reduce the number of symbols. PAL III's symbol capacity is:

Using ASR33; 655 symbols. The basic symbol table contains 65 symbols (see Appendix) leaving 59Ø user-defined symbols. Using the High-Speed Reader; 56Ø symbols. The basic symbol table contains 65 symbols leaving 495 user-defined symbols.

EXPRESSIONS

Symbols and numbers are combined with certain operators to form expressions. There are three operators:

+	plus	this signifies 2's complement addition
-	minus	this signifies 2's complement subtraction
␣	space	space is interpreted in context. Since a PDP-8 instruction has an operation code of three bits as well as an indirect bit, a page bit, and seven address bits, the Assembler must combine memory reference instructions

in a manner somewhat different from the way in which it combines operate or IOT instructions. The Assembler accomplishes this by differentiating the symbols in its permanent symbol table. The following symbols are used as memory reference instruction op codes:

AND	ØØØØ	logical AND
TAD	1ØØØ	Two's complement ADdition
ISZ	2ØØØ	Index and Skip if Zero
DCA	3ØØØ	Deposit and Clear Accumulator
JMS	4ØØØ	JuMp to Subroutine
JMP	5ØØØ	JuMP
FADD	1ØØØ	Floating ADDition
FSUB	2ØØØ	Floating SUBtraction
FMPY	3ØØØ	Floating MultiPlY
FDIV	4ØØØ	Floating DIVide
FGET	5ØØØ	Floating GET
FPUT	6ØØØ	Floating PUT
FNOR	7ØØØ	Floating NORmalize
FEXT	ØØØØ	Floating EXiT

When the Assembler has processed one of these symbols, the space acts as an address field delimiter:

```
    *41ØØ
    JMP  A
A,   CLA
```

A has the value 41Ø1, JMP has the value 5ØØØ, and the space acts as a field delimiter. These symbols are combined as follows:

```
A    1ØØ ØØ1 ØØØ ØØ1
JMP  1Ø1 ØØØ ØØØ ØØØ
```

The seven address bits of A are taken, i.e.:

ØØØ ØØ1 ØØØ ØØ1

The remaining bits of the address are tested to see if they are zero's (page zero reference); if they are not, the current page bit is set:

ØØØ Ø11 ØØØ ØØ1

The operation code is then ORed into the expression to form:

1Ø1 Ø11 ØØØ ØØ1

or, written more concisely:

53Ø1

In addition to the above outlined tests, the page bits of the address field are compared with the page bits of the current location counter. If the page bits of the address field are nonzero and do not equal the page bits of the current location counter, an out-of-page reference is being attempted and the Illegal Reference diagnostic is printed on PASS2 or PASS3.

For example:

```
*41ØØ
A,    CLA CLL
        .
        .
        .
*72ØØ
      JMP A
```

The symbol in the address field of the jump instruction has a value of 41ØØ while the current location counter, i.e., the address where the instruction will be placed in memory, has a value of 72ØØ. This instruction is illegal on the PDP-8 and will be flagged during PASS2 or PASS3 by the Illegal Reference diagnostic:

```
IR   41ØØ   AT   72ØØ
```

The value 53ØØ would be assembled at location 72ØØ.

The symbol I caused the indirect bit (bit 3) to be set in a memory reference instruction: For example:

```
DCA   I   1Ø
```

would produce:

```
Ø11   1ØØ   ØØ1   ØØØ
```

or:

```
341Ø
```

When a space occurs in an expression that does not contain a memory reference instruction op code, it means inclusive OR:

For example:

```
CLA   CLL
```

the symbol CLA has a value of 72ØØ and the symbol CLL has a value of 71ØØ; CLA CLL would produce 73ØØ. User-defined symbols are treated as nonmemory reference instructions (see Pseudo-Instructions).

For example:

```
A=333
*222
B, CLA
```

Then the expressions and their values are shown below:

```
A+B       Ø555
A-B       Ø111
A␣B       Ø333
-A        7445
1-B       7557
B-1       Ø221
-71       77Ø7
etc.
```

An expression is terminated by either a carriage-return (↵) or a semicolon (;). If any information was generated to be loaded, the current location counter is incremented.

Example:

```
RAR; RTR; CMA↵
```

Produces three registers of information and the current location counter is incremented after each expression. The statement:

```
HALT=HLT CLA↵
```

produces no information to be loaded (it produces an association in the Assembler's symbol table) and hence does not increment the current location counter.

```
      *4721
TEMP,↵
TEM2, Ø↵
```

The current location counter is not incremented after the line TEMP,↵ and hence the two symbols TEMP and TEM2 are assigned the same value, in this case 4721.

CURRENT ADDRESS INDICATOR

The single character period (.) has, at all times, a value equal to the value of the current location counter. It may be used as any number or symbol (except to the left of the equal sign).

Example:

```
*2ØØ
JMP    .+2
```

is equivalent to JMP 2Ø2.

```
        *3ØØ
         .+24ØØ
```

would produce, in register 3ØØ, the quantity 27ØØ

Example:

```
        *22ØØ
 CALL=JMS 1  .
        27
```

Since the second line, CALL=JMS 1 . does not increment the current location counter, ØØ27 would be placed in register 22ØØ and CALL would have the value of 1ØØ 11Ø $ØØØ_2$ or $46ØØ_8$.

The properties of the character (.) have been slightly changed; so that, it now acts as a terminator. Previously, PAL III would neither diagnose nor correctly assemble expressions such as: JMP. (where there is no space between the P and the .) PAL III now treats this (JMP.) as if it were this (JMP .)

COMMENTS

A comment field is indicated by the slash (/) character. The Assembler will ignore everything from the slash to the next carriage return.

Example:

```
        CLA                /THIS IS A COMMENT
```

PSEUDO-INSTRUCTIONS

There are several pseudo-instructions that are used to direct the Assembler. These are:

DECIMAL	Set the current radix to decimal
OCTAL	Set the current radix to octal
PAUSE	Stop the Assembler. The current pass is not terminated. PAUSE must be at the physical end of the program tape as the reader routines are buffered and the buffer is emptied when PAUSE is detected. The assembly is continued by depressing CONTINUE. Two or more tapes must be used with the PAUSE instruction.
FIELD	Causes a field setting to be punched during PASS2. This is recognized by the Binary Loader (DEC-08-LBAA-PM) and causes all subsequent information to be loaded into the field specified by the expression. The expression must be between Ø and 7, inclusive.
EXPUNGE	Erase the entire symbol table except for the pseudo-instructions.
FIXTAB	Fix the current symbol table. Symbols that have been fixed are not printed in the symbol table at the end of PASS1 or PASS3.

FIXMRI — Fix memory reference instruction. This may be given only after EXPUNGE. It tells the Assembler that the following symbol definition is a memory reference instruction and is to be treated as described under Expressions.

Example:

```
EXPUNGE
FIXMRI TAD=1ØØØ
FIXMRI DCA=3ØØØ
CLA=72ØØ
FIXTAB
PAUSE
```

When this program segment is read into the Assembler during PASS1, all symbol definitions are deleted and the three symbols listed are added to the table.

This process is often performed to alter the Assembler's symbol table so that it contains only those symbols that will be used. This may increase the Assembler's capacity for other user-defined symbols.

I — Symbolic representation for indirect addressing.

Example:

```
DCA I STORE
```

Z — Optional method of denoting a Page Ø reference.

Example:

```
DCA Z ADD
```

CHAPTER 4

PROGRAM PREPARATION AND ASSEMBLER OUTPUT

The source language tape (symbolic tape) is prepared in ASCII code on 8-channel punched paper tape using an off-line Teletype or the on-line Symbolic Tape Editor (DEC-08-ESAB-D). In general, a program should begin with leader code which may be blank tape, code 2ØØ, or rubouts.

PROGRAM TAPE

Since the Assembler ignores certain codes, these may be used freely to produce a more readable symbolic source tape. These codes are tab, line-feed, and form-feed.

The Assembler will also ignore extraneous spaces, carriage-return/line-feed combinations, and blank tape.

The program body consists of statements and pseudo-instructions. The program is terminated by the dollar sign followed by some trailer code. If the program is large, it may be segmented by use of the pseudo-instruction PAUSE. This often facilitates the editing of the source program since each section will be physically smaller.

The Assembler initially sets its current location counter to Ø2ØØ. This is reset whenever the asterisk is processed.

During PASS1, all illegal characters cause a diagnostic to be printed. The character is ignored.

The following two programs are identical:

```
*2ØØ
/EXAMPLE OF FORMAT
/GENERATOR
BEGIN, Ø/START OF PROGRAM
KCC
KSF/WAIT FOR FLAG
JMP.-1/FLAG NOT SET YET
KRB/READ IN CHARACTER
DCA CHAR
TAD CHAR
TAD MSPACE/IS IT A SPACE?
SNA CLA
HLT/YES
JMP BEGIN+2 /NO: INPUT AGAIN
CHAR, Ø/TEMPORARY STORAGE
MSPACE, -24Ø/-ASCII EQUIVALENT
/END OF EXAMPLE
$
```

```
*2ØØ
/EXAMPLE OF FORMAT
/GENERATOR
BEGIN,      Ø              /START OF PROGRAM
            KCC
            KSF            /WAIT FOR FLAG
            JMP.-1         /FLAG NOT SET YET
            KRB            /READ IN CHARACTER
            DCA CHAR
            TAD CHAR
            TAD MSPACE     /IS IT A SPACE?
            SNA CLA
            HLT            /YES
            JMP BEGIN+2    /NO: INPUT AGAIN
CHAR,       Ø              /TEMPORARY STORAGE
MSPACE,     -24Ø           /-ASCII EQUIVALENT
/END OF EXAMPLE
$
```

Both of these programs are identical and produce the same binary code. The second, however, is easier to read.

During PASS1, the Assembler reads the source tape and defines all symbols used. The user's symbol table is printed (or punched) at the end of PASS1. If any symbols remain undefined, the UA diagnostic is printed. The symbol table is printed in alphabetic order. If the program listed above were assembled, the PASS1 output would be:

```
BEGIN       Ø2ØØ
CHAR        Ø213
MSPACE      Ø214
```

During PASS2, the Assembler reads the source tape and generates the binary code using the symbol table equivalences defined during PASS1. The binary tape that is punched may be loaded by the Binary Loader (DEC-08-LBAA-PM). This binary tape consists of leader code, an origin setting, and then data words. Every occurrence of an asterisk experssion causes a new origin to be punched on the tape and resets the Assembler's current location counter. At the end of PASS2, the checksum is punched on the binary tape and trailer code is generated. During PASS2, the Assembler may diagnose an Illegal Reference. When using the ASR33 Punch, the diagnostic will be both typed and punched and will be preceded and followed by rubouts. The Binary Loader will ignore everything that has been punched on a tape between rubouts.

During PASS3, the Assembler reads the source tape and generates the code from the source statements. The assembly listing is typed (or punched). It consists of the current location counter, the generated code in octal, and the source statement. The symbol table is typed at the end of the pass. If the program listed above were assembled, the PASS3 output would be:

```
                        *2ØØ
                        /EXAMPLE OF FORMAT
                        /GENERATOR
Ø2ØØ      ØØØØ          BEGIN,   Ø                /START OF PROGRAM
Ø2Ø1      6Ø32                   KCC
Ø2Ø2      6Ø31                   KSF              /WAIT FOR FLAG
Ø2Ø3      52Ø2                   JMP.-1           /FLAG NOT SET YET
Ø2Ø4      6Ø36                   KRB              /READ IN CHARACTER
Ø2Ø5      3213                   DCA CHAR
Ø2Ø6      1213                   TAD CHAR
Ø2Ø7      1214                   TAD MSPACE       /IS IT A SPACE?
Ø21Ø      765Ø                   SNA CLA
Ø211      74Ø2                   HLT              /YES
Ø212      52Ø2                   JMP BEGIN+2      /NO: INPUT AGAIN
Ø213      ØØØØ          CHAR,    Ø                /TEMPORARY STORAGE
Ø214      754Ø          MSPACE,  -24Ø             /-ASCII EQUIVALENT
                        /END OF EXAMPLE

BEGIN     Ø2ØØ
CHAR      Ø213
MSPACE    Ø214
```

CHAPTER 5

OPERATING INSTRUCTIONS

The PAL III Assembler is provided as a binary tape. This is loaded into the PDP-8 memory by means of the Binary Loader, using either the ASR33 Reader or the High-Speed Reader (see DEC-08-LBAA-D). The Assembler will use either the ASR33 Reader or the high-speed reader to read the source language tape, and it will use either the ASR33 Punch or the High-Speed Punch for output. The selection of I/O devices is made by the Assembler when it is started. The source language tape must be in the proper reader, with the reader and punch turned on. When using the high-speed punch, the symbol table will be typed on the ASR33 if bit 11 of the switch register is 0 (down); it will be punched on the high-spped punch if bit 11 of the switch register is a 1 (up). When using the high-speed punch, the symbol table output, the telepunch should be left on, since the symbol table produced may be read by DDT (see DEC-08-CDDA-D). All diagnostics will be typed on the ASR33 (except for the undefined address diagnostic when using the high-speed punch and the bit 11 switch option). The binary tape produced during PASS2 will be punched using the ASR33 punch or the High-Speed Punch if it is included in the machine configuration and turned on. The only diagnostic in PASS2 will be Illegal Reference. Since this is typed on the ASR33, it may also be punched on the binary tape. It will, however, be ignored by the Binary Loader. The bit 11 switch option may be used during PASS3 also. If the machine is not equipped with a High-Speed Punch, bit 11 will have no effect.

In addition to the binary tape of the Assembler, the user is provided with an ASCII tape containing symbol definitions for the instruction sets of the available options to the PDP-8 (i.e., card readers, magnetic tapes, A/D converters). Since there is only a finite amount of space available, expanding the number of permanent symbols that the Assembler recognizes decreases the maximum number of symbols the user may have available. For this reason, the ASCII Extended Definitions tape should be edited to contain definitions for only those options which the user has acquired. This tape should be read into the Assembler only on PASS1. Since it permanently fixes the symbols it contains, it should not be read again until PAL III is reloaded.

1. Load the Assembler using either the ASR33 Reader or the High-Speed Reader.
2. Set Ø2ØØ into the switch register; press LOAD ADDRESS.
3. Place the source language tape in the reader. Turn the reader on; turn the punch on.

4. Set Bits Ø and 1 of the switch register for the proper pass. These settings are:

Bit Ø	Bit 1	
Ø	1	PASS1
1	Ø	PASS2
1	1	PASS3

PASS1 is required so that the Assembler can initialize its symbol table and define all user symbols. After PASS1 has been made, either PASS2 or PASS3 may be made.

5. Bit 11 switch option

During PASS1 Bit 11 = 1	Punch symbol table on high-speed punch if it is in the machine configuration.
Bit 11 = Ø	Type (and punch) the symbol table on the ASR33.
During PASS2	No effect
During PASS3 Bit 11 = 1	Punch assembly listing tape, in ASCII, on high-speed punch.
Bit 11 = Ø	Type assembly listing on ASR33.

6. Press START. The Assembler will halt at the end of each pass. Proceed from step 3. If the Assembler has halted because of a PAUSE statement, put the next tape into the reader and press CONTINUE.

SUMMARY

PASS1

The Assembler reads the source tape, defines all user symbols, and outputs the user symbol table in alphabetic order. PASS1 diagnostics are:

IC dddd AT xxxx   Illegal Character

where dddd is the value of the illegal character and xxxx is the value of the current location counter when the character was processed. The character is ignored.

RD XXXXXX AT dddd   ReDefinition

where XXXXXX is the symbol being redefined and dddd is the value of the current location counter at the point of redefinition. The symbol is redefined.

DT XXXXXX AT dddd   Duplicate Tag

An attempt is being made to redefine a symbol using the comma. XXXXXX is the symbol and dddd is the value of the current location counter. The previous value of the symbol is retained and the symbol is not redefined.

ST XXXXXX AT dddd   Symbol Table full

where XXXXXX is the symbol causing the overflow and dddd is the value of the Current Location Counter at the point of overflow. The Assembler halts and may not be restarted.

UA XXXXXX AT dddd   Undefined Address

where XXXXXX is the symbol that was used, but never defined, and dddd is the value of the Current Location Counter when the symbol was first processed. This is typed with the symbol table at the end of PASS1. The symbol is assigned a value equal to the highest address on the memory page where it was first used.

PASS2

The Assembler reads the source tape and using the symbol table defined during PASS1, generates and punches the binary code. This binary tape may then be loaded by the Binary Loader. The PASS2 diagnostic is:

IR dddd AT xxxx   Illegal Reference

where dddd is the address being referenced and xxxx is the value of the Current Location Counter. The illegal address is then treated as if it were on the proper memory page.

Example:

```
*73Ø6
 JMP   3Ø7
```

would produce:

```
IR Ø3Ø7 AT 73Ø6
```

and would generate 53Ø7 to be loaded into location 73Ø6.

PASS3 The Assembler reads the source tape and, using the symbol table defined during PASS1 generates and types the code represented by the source statements. The Current Location Counter, the contents, and the source statement are typed side by side on one line. If bit 11 of the switch register is a 1 and the machine configuration includes the high-speed punch, the assembly listing will be punched in ASCII. The PASS3 diagnostic is Illegal Reference.

CHAPTER 6

SYMBOL TABLE ALTERATION

PAL III contains a table of symbol definitions for the basic PDP-8 and its most common optional peripheral devices. These are the symbols such as TAD, RFC or SPA, which do not have to be defined in every program. This table is considered to be PAL III's permanent symbol table. All the symbols it contains are listed under the heading BASIC SYMBOLS in Appendix 1 of this manual. If the user had purchased one or more of the optional devices whose instruction set is not defined among the BASIC SYMBOLS, for example, EAE or an A/D CONVERTER, it would be desirable if he could add the necessary symbol definitions to the permanent symbol table. This would eliminate the need for him to define these symbols in every program he writes. The opposite case would be the user who needs more space for his symbols. He would like to be able to delete all definitions except the ones he will actually use in his program.

For such purposes PAL III has three pseudo-instructions that may be used to alter its permanent symbol table. These pseudo-instructions are recognized by the Assembler only during PASS1. During either PASS2 or 3, they are ignored and have no effect.

The pseudo-instructions that alter the symbol table are:

EXPUNGE — Erase the entire permanent symbol table, except for the 9 pseudo-instructions listed in Appendix 1 under BASIC SYMBOLS.

FIXMRI — Fix Memory Reference Instructions. This must be followed on the same line by a symbol definition statement (parameter assignment) since the memory reference instructions are constructed in the symbol immediately following the pseudo-instructions. In other words the letters FIXMRI must be followed by one space, the symbol for the MRI to be defined, an equal sign, and the actual value of the symbol to the immediate left of the equal sign. The pseudo-instruction must be repeated for each MRI to be defined. All MRI's must be defined before the definition of any other symbol.

EXAMPLE:
```
EXPUNGE
FIX MRI    TAD = 1000
FIX MRI    DCA = 3000
```

FIXTAB — FIX the current symbol TABLE. All symbols that have been defined before the occurance of this pseudo-instruction are made part of the permanent symbol table and will not be printed in the symbol table at the end of PASS1 or PASS3.

An actual tape to add two symbols to those already in PAL III's permanent symbol table would have punched on it in ASCII:

```
CDF=62Ø1
CIF=62Ø2
FIXTAB
PAUSE
```

To use such a tape the user would:

1. Read in PAL III with the Binary Loader.
2. Set 200 in the SWITCH REGISTER and press LOAD ADDRESS.
3. Set switches for PASS1.
4. Put definitions tape (ASCII) in the proper reader.
5. Press START.

The PAUSE pseudo-instruction at the end of the tape indicates to the Assembler that the current PASS is not ended and another tape is to follow.

6. With switches still set to PASS1, put user's program in reader and press CONTINUE on the console.

The next program to be assembled should not be preceded by the definitions since they are already in the permanent symbol table and will be there until PAL III is reloaded.

After altering the symbol table to fit his needs the user might wish to keep PAL III in this state. This can be done by punching a binary of the section of core occupied by PAL with its new symbol table.

To do this:

1. Read in PAL III and modify symbol table as desired.
2. PAL III's symbol table begins at location 2350_8. Count all the symbols in the altered symbol table. Since each symbol and its value require four registers, multiply this number by 4. Convert this number to octal and add it to 2350_8. This number is the upper limit of PAL III The lower limit is 0001.

3. Using the directions for Binary Punch Routine. (Digital-8-5-U) and the limits as stated in 2 above punch out the PAL III Assembler itself.

4. The output of the Binary Punch Routine is the Assembler with the modified Symbol Table and may be loaded with the binary loader.

EXAMPLE: PAL III is loaded.

The following ASCII tape is read in on PASS1:

CDF = 6201
CIF = 6202
RDF = 6214
RIF = 6224
RMF = 6244
RIB = 6234
FIXTAB
PAUSE

The Assembler now has in its symbol table the "MEMORY EXTENSION CONTROL" symbols and definitions. Six symbols were added and none removed. There were 84 symbols in the basic Assembler, there are now 90 symbols which require a total of $360_{(10)}$ or 550_8 locations. Since the symbol table starts at 2350, it extends to $2350_8 + 550_8$ or 3120_8. The Binary Punch Routine is used to punch from 0001_8 through 3120_8 and the output is the Assembler with all the basic symbols plus memory extension symbols.

APPENDIX 1

SYMBOL LISTS

BASIC SYMBOLS

/PSEUDO INSTRUCTIONS
FIELD
EXPUNGE
FIXMRI
PAUSE
FIXTAB
DECIMAL
OCTAL
I
Ƶ

/MEMORY REFERENCE INSTRUCTIONS

AND	ØØØØ
TAD	1ØØØ
ISƵ	2ØØØ
DCA	3ØØØ
JMS	4ØØØ
JMP	5ØØØ

/PROGRAM INTERRUPT

ION	6ØØ1
IOF	6ØØ2

/HIGH-SPEED READER

RSF	6Ø11
RRB	6Ø12
RFC	6Ø14

/HIGH-SPEED PUNCH

PSF	6Ø21
PCF	6Ø22
PPC	6Ø24
PLS	6Ø26

/KEYBOARD/READER

KSF	6Ø31
KCC	6Ø32
KRS	6Ø34
KRB	6Ø36

/GROUP 2 OPERATES

HLT	74Ø2
OSR	74Ø4

/FLOATING-POINT INSTRUCTIONS

FEXT	ØØØØ
FADD	1ØØØ
FSUB	2ØØØ
FMPY	3ØØØ
FDIV	4ØØØ
FGET	5ØØØ
FPUT	6ØØØ
FNOR	7ØØØ

/TELEPRINTER/PUNCH

TSF	6Ø41
TCF	6Ø42
TLS	6Ø46
TPC	6Ø44

/GROUP 1 OPERATES

NOP	7ØØØ
IAC	7ØØ1
RAL	7ØØ4
RTL	7ØØ6
RAR	7Ø1Ø
RTR	7Ø12
CML	7Ø2Ø
CMA	7Ø4Ø
CLL	71ØØ
CLA	72ØØ

/COMBINED OPERATES

CIA	7Ø41
LAS	76Ø4

SKP	741Ø	STL	712Ø
SNL	742Ø	GLK	72Ø4
SƵL	743Ø	STA	724Ø
SƵA	744Ø		
SNA	745Ø		
SMA	75ØØ		
SPA	751Ø		

/DECTAPE DUAL TRANSPORT TYPE 555 AND CONTROL TYPE 552

MMMM	6757	MMSF	6761
MMMF	6756	MMCF	6772
MMML	6766	MMSC	6771
MMLS	6751	MMRS	6774
MMLM	6752	MMCC	6762
MMLF	6754	MMLC	6764

/DECTAPE TRANSPORT TYPE TU55 AND CONTROL TYPE TCØ1

DTRA	6761	DTSF	6771
DTCA	6762	DTRB	6772
DTXA	6764	DTLB	6774

/MEMORY PARITY TYPE 188

SMP	61Ø1
CMP	61Ø4

EXTENDED SYMBOLS

/PDP -5 EAE SYMBOLS 153*

CAM	61Ø1	SZO	6114
LMQ	61Ø2	DIV	6121
LAR	61Ø4	RDM	6122
MUL	6111	SAF	6124
RDA	6112		

/PDP-8 EAE SYMBOLS 182

MUY	74Ø5	ASR	7415
DVI	74Ø7	LSR	7417
NMI	7411	MQL	7421
SHL	7413	SCA	7441
MQA	75Ø1	CAM	7621

/MEMORY EXTENSION CONTROL TYPE 183

CDF	62Ø1	RIF	6224
CIF	62Ø2	RMF	6244
RDF	6214	RIB	6234

/AUTO RESTART TYPE KRØ1

SPL = 61Ø2

* PDP-5 EAE symbol definitions do not appear on the actual tape due to a conflict in the CAM instructions of PDP-5 and PDP-8. PDP-8 EAE symbols should be deleted if those for PDP-5 are inserted in the extended symbols tape.

/AD CONVERTER TYPE 189

ADC	6ØØ4

/AD CONVERTER/MULTIPLEXER 138E/139E

ADSF	6531	ADCC	6541
ADCV	6532	ADSC	6542
ADRB	6534	ADIC	6544

/OSCILLOSCOPE DISPLAY TYPE 34D

DCX	6Ø51	DYL	6Ø63
DXL	6Ø53	DIX	6Ø54
DCY	6Ø61	DIY	6Ø64
DXS	6Ø57	DYS	6Ø67

/SCOPE TYPE 30N

DLB	6Ø74

/LIGHT PEN TYPE 37Ø

DSF	6Ø71	DCF	6Ø72

/PLOTTER AND CONTROL TYPE 35ØB

PLSF	65Ø1	PLCF	65Ø2
PLPU	65Ø4	PLPR	6511
PLPU	6512	PLDD	6514
PLPL	6521	PLUD	6522
PLPD	6524		

/CARD READER AND CONTROL TYPE CRØ1C

RCSF	6631	RCSP	6671
RCRA	6632	RCSE	6671
RCRB	6634	RCRD	6674

/CARD READER TYPE 451

CRSF	6632	CERS	6634	/also services card punch 450
CRRB	6671	CRSA	6672	
CRSB	6674			

/CARD PUNCH AND CONTROL TYPE 45Ø

CPSF	6631	CPSE	6642
		CPLB	6644
CPCF	6641	/CERS	as appears under card reader 451

/LINE PRINTER TYPE 645

LCF	6652	LPR	6655
LSF	6661	LCB	6662
LLB	6664		

/SERIAL DRUM 25Ø AND 251

DRCR	66Ø3	DRCW	66Ø5
DRCF	6611	DREF	6612
DRTS	6615	DRSE	6621
DRSC	6622	DRCN	6624

/MAGNETIC TAPE TYPE 57A

MSCR	67Ø1	MCD	67Ø2
MTS	67Ø6	MSUR	6711
MNC	6712	MTC	6716
MSWF	6721	MDWF	6722
MCWF	6722	MEWF	6722
MIWF	6722	MSEF	6731
MDEF	6732	MCED	6732
MEEF	6732	MIEF	6732
MTRS	6734	MCC	6741
MRWC	6742	MRCA	6744
MCA	6745		

/MAGNETIC TAPE TYPE 58Ø

TSRD	6715	TIFM	67Ø7
TSWR	6716	TSDF	6721
TSSR	6722	TSST	6724
TWRT	6731	TCPI	6732
TSRS	6734		

/EIGHT CHANNEL SAMPLE AND HOLD CONTROL TYPE ACØ1A

/OPTION TO TYPE 139E MULTIPLEXOR

HSC	6571
HAC	6572
SAC	6574

/DATA COMMUNICATION SYSTEMS TYPE 63Ø

TTINCR	64Ø1	TTRL	6414
TTI	64Ø2	TTSKP	6421
TTO	64Ø4	TTXON	6422
TTCL	6411	TTXOF	6424
TTSL	6412		