The SAPDP Program Set for Sigma 5 Assembly

D. E. Erickson
Communications Systems Research Section

This article describes a set of programs that have been written to enable the Sigma 5 computer to assemble programs for the PDP-11 minicomputer. It consists of two parts: a system procedure deck, which allows SIGMA METASYMBOL to assemble a source language similar to PDP's own PAL-11; and a secondary loader, which reformats the Sigma 5 load module into PDP-11 absolute binary format and punches it onto paper tape. The syntactic differences between this assembler and PAL-11 are described, as well as the process of generating a PDP-11 program using this program set on the Sigma 5.

I. Introduction

The Sigma assembler for the PDP (SAPDP) is a set of programs for the Sigma 5 computer which can be used to build programs for the PDP-11 computer. It consists of two parts: a system procedure deck, which allows METASYMBOL to assemble a source language similar to PAL-11; and a secondary loader, which reformats the Sigma 5 load module into PDP-11 absolute loader format and punches it onto paper tape. Figure 1 of the article by C. C. Klimasauskas[1] is a good description of the use of SAPDP if references to X930 within that figure are replaced by references to SAPDP.

The procedure deck defines the valid operators to the SIGMA 5 METASYMBOL assembler, and determines what code will be generated for the valid source statements. METASYMBOL procedures are similar to macro definitions. The code produced from the source program under control of the procedures is formatted by METASYMBOL into a Sigma relocatable object module (ROM) containing relocation information, external references and definitions, and generated code.

A number of ROMs may then be linked together, and the external references and definitions resolved by the Sigma loader. Normally the loader gives the user the option of saving relocation information, creating a task-control block (TCB), and satisfying unresolved external references from the system library. These are Sigma-oriented functions and should be disallowed during loading for PDP-11 programs by specifying the options (ABS), (NOTCB), and (NOSYSLIB) on the load control card. The Sigma loader also has the capability of relocating the

[1]See "The X930 Program Set for Sigma 5 Assembly" by C. C. Klimasauskas in this issue.

program to any boundary which is a multiple of 200 hex. It will automatically relocate to the background lower limit unless the BIAS option is specified on the load card. (BIAS, 0) will cause the first ROM to be not relocated. The Sigma loader structures its output into a file called a load module (LMN), which consists of the core image program and several records of control information. The Sigma 5 has write protection, so the core image is in several pieces, one for each protection type.

The secondary loader reads a Sigma load module and writes the 00 protection-type core image data in the format which is loaded by the PDP-11 absolute loader. So that the program need not start on a multiple of 200 hex, the secondary loader skips all data until the first non-zero 16-bit word. Thus the first valid word in the PDP-11 program must be non-zero. At present, output from the secondary loader is to paper tape, but when a direct link is established between the Sigma 5 and PDP-11 computers, the output could be sent across this link.

II. The Source Language

The source language is defined by the system procedure deck SYSTEM PDP-11. Although similar to PAL-11, it has many significant differences made necessary by the required format for METASYMBOL statements.

A. Syntax

This discussion is intended to enable the programmer familiar with the PDP-11 Paper Tape Programming Handbook to write source code for SAPDP. A programmer who is also familiar with the XDS Sigma 5/7 Symbol/Metasymbol Reference Manual, hereinafter referred to as the METASYMBOL manual, may skip down to the section titled modes of addressing.

1. Characters

Alphabetic:	A through Z,$,@,#,:, and _ (break character, underscore).
Numeric:	0 through 9.
Special:	blank,(,),+,−,&,|,' (single quote),*, and comma.

2. Symbols. Symbols may consist of 1–63 alphanumeric characters, at least one of which is alphabetic. The special symbols $ and $$ stand for the values of the location counters and their use is described later. As in the PAL-11 language, blanks may not be embedded in a symbol, but, unlike PAL-11, all of the characters in a symbol are significant, not just the first six.

3. Constants. The types of constants of interest to the PDP-11 programmer are as follows:

Decimal integer: a string of numeric digits (not followed by a decimal point)

Octal constant: a string of octal digits surrounded by quotes and preceded by the letter O. Example: O'1777'

Character string constant: A string of non-quote characters surrounded by quotes and optionally preceded by the letter C. To represent a quote in a character string, one may use two consecutive single quotes. Example: C'AB''C' is the string AB'C

4. Expressions. Expressions are composed of terms and operators. The terms may be symbols or constants, and may be forward references to non-redefinable symbols. Many operators are available. They are described in the METASYMBOL manual. Four of them correspond to the PAL-11 operators: +,−,& (logical and), and | (logical inclusive or).

Parentheses may be used to force the order of operations. In the absence of parentheses, the order is set by the binding strength of the operators described in the METASYMBOL manual. There is no default term or operator.

5. Statements. As in PAL-11, a SAPDP statement is composed of up to four fields which are identified by their order of appearance. These fields are: LABEL OPERATOR OPERAND COMMENT.

Termination of the fields is somewhat different, however, due in part to the fact that input is from cards rather than paper tape. A statement begins in column 1 of a card and ends on column 72. Any of the fields may be terminated by the end of statement (EOS), and all but the COMMENTS field may be terminated by a blank. A semicolon in any but the COMMENTS field will continue the statement beginning with the first non-blank character of the next card, which must be blank in column 1. The semicolon may not be between quotes.

The LABEL field begins in the first column of the statement and terminates with the first blank or end of statement. If the first character of the statement is blank, the LABEL is not present. If present, the LABEL may be any symbol except $ or $$.

The OPERATOR begins with the first non-blank character following the LABEL field, and terminates with a

blank or EOS. The OPERATOR is either an assembler directive or an instruction mneumonic.

The OPERAND begins with the first non-blank after the OPERATOR and terminates with a blank or EOS. The form of the OPERAND depends on the OPERATOR.

The COMMENTS field extends from the end of the OPERAND to the end of the statement.

B. Modes of Addressing

One of the major differences in the source languages is the method of representing the modes of addressing. SIGMA 5 METASYMBOL has special symbol conventions which necessitate the alteration of address mode syntax. First, METASYMBOL provides no special way of declaring a given symbol to represent a register, so SAPDP assumes that any expression which has a value from 0 to 7 represents a register. Such an expression will be referred to as a register expression (RE), any other expression as an expression (E). SAPDP reserves four special symbols to aid in describing addressing modes: @, @D, @I, and #. Table 1 describes the syntax.

The last entry in the table illustrates the method of specifying deferred addressing; merely prefix the address with the character * instead of @ as one would for the PAL-11 assembler. Example of deferred auto increment: *(@I,RE).

C. Instruction Mnemonics

All of the instruction mneumonics described in Appendix B of the PDP-11 Paper Tape Software Programming Handbook are available unaltered, with the exception of COM which has been changed to COMW due to a conflict with a METASYMBOL directive.

D. Assembler Directives

The syntax of the directives has been changed slightly. The .EOT directive is no longer necessary and has been dropped. The period preceding the names of the other directives has been eliminated. Thus .END is now written END.

E. Assignment of Symbol Values

To define and assign a value to a symbol, it must appear in the LABEL field of a statement. If the OPERATOR is an instruction mneumonic or a BYTE, WORD, RES, or EVEN directive, the LABEL is not redefinable and is assigned the current value of the location counter $. If it is an EQU or SET directive, the LABEL is assigned the value of the OPERAND expression. This is similar to direct assignment in PAL-11. A SET symbol may be redefined, whereas an EQU symbol may not. Forward references may not be made to redefinable symbols. Examples:

```
A  SET  B     a forward reference to B
B  EQU  5     B is not redefinable
A  SET  A+1   A is redefinable
```

F. The Location Counters

SAPDP has two location counters: the load location counter $$, and the execution location counter $. These cannot be assigned values directly as the location counter can in the PAL-11 assembler. This is due to the restriction that $ and $$ cannot be used as labels. Space for data can be reserved, however, by using the reserve directive

```
LABEL RES,1  EXPRESSION
```

to replace

```
LABEL =.
    .=.+EXPRESSION
```

where EXPRESSION evaluates to a positive value indicating the number of bytes to be reserved. LABEL is optional. The symbol $ in SAPDP can be used in expressions wherever . was used in PAL-11. Example:

```
BR  $+5
```

The initialization of the location counter $ for a program which is to be loaded for execution at octal location SSSS is done by the directive

```
ORG,1  O'SSSS'
```

as the first card in the deck. Unless one is familiar with METASYMBOL and this particular application of it, attempts to alter the location counters others than by the methods described here could easily lead one astray.

G. The Program Deck

The SAPDP program deck must begin with two directives establishing the environment:

```
ORG,1  O'SSSSS'
SYSTEM PDP11
```

where SSSSS is the octal address of the first byte of the program. The first word of the program must be non-zero.

The END directive should be the last card in the program deck. If the OPERAND field of the END directive is a non-zero address, it will become the automatic starting address of the program, and the PDP-11 absolute loader will transfer control to that address. If there is no OPERAND or it is zero, a transfer address of 1 is generated, causing the PDP-11 loader to halt after loading the program.

H. Extended Features

All of the features of SIGMA METASYMBOL described in the METASYMBOL manual are available to the SAPDP user. These include the ability to define PROCs and perform conditional assembly, to create a compressed source deck, to generate a concordance of symbols used, and many other features. One may also use the features of the Sigma 5 loader, such as linking of external references and definitions, and creation and use of user libraries of preassembled subroutines.

In addition, two PROCs define directives which generate floating point format data. The directives are FSC (Floating Short Constant), and FLC (Floating Long Constant). These are used with types of constants not described above as operands:

Floating Short Constant FS'CCCCC'
Floating Long Constant FL'CCCCCC'

where CCCCC represents a number of the following form:

.D ,D ,D.D

where D is a decimal string, optionally preceded by a sign and optionally followed by a decimal exponent composed of the letter E followed by an optional sign and one or two decimal digits.

The FSC directive takes the short operands and the FLC the long operands. Examples:

```
FSC   FS'1.5',FS'78E-5'   Generates 3 words for 1.5
                          and 3 for 0.00078
FLC   FL'36.245E24'
```

If one wishes to create a new control section, either within a METASYMBOL assembly by means of a CSECT directive, or in a separate assembly, he must follow certain conventions. First, he must make sure that the location counters are byte addressing, and, second, he must be sure to initialize them correctly. He accomplishes both of these objectives with an ORG directive:

ORG,1 O'SSSSS'

If the CSECT is the first CSECT that the loader will see, SSSSS should be the load address for the paper tape. If not, SSSSS should be 0 and this CSECT will be automatically loaded following the preceding one. External references and definitions may be used and will be resolved by the Sigma 5 loader, so commonly used subroutines may be preassembled and saved on cards or rad.

III. The Secondary Loader

The secondary loader reformats Sigma 5 load modules into PDP-11 absolute loader format. Input to the secondary loader is through the M:EI DCB and output is through the M:PO DCB. The input load module is typically on a RAD file, and the output is to paper tape. The M:EI and M:PO DCBS must be assigned to the appropriate files or devices before execution of the secondary loader. The PDP-11 absolute loader expects control information at the beginnings and ends of the records it reads, and the secondary loader supplies this.

In addition, the addressing schemes of the Sigma 5 and the PDP-11 are somewhat different. Within a PDP-11 sixteen-bit word, the even-numbered byte contains the least significant part of the word and the next higher odd byte contains the most significant part of the word. In the Sigma 5 this is reversed. Thus the secondary loader must reverse each pair of bytes. Unfortunately, this means that it also erroneously reverses bytes which were generated to be loaded at a particular byte address; therefore, the BYTE directive has been written to reverse the bytes when generating them to be consistent with word generation. A METASYMBOL DATA,1 directive should not be used for byte items for this reason.

The length of a core image segment in a Sigma load module is a multiple of a Sigma double word (64 bits) so the last byte of a program created through SAPDP will load on a PDP-11 address which is a multiple of 8 less 1. This means that several bytes of zeros may follow the actual program.

IV. Control Cards

The following is the deck setup for assembling in a program under SAPDP and punching a tape for the PDP-11.

```
!JOB PDP,PDP
!METASYM SI,LO,GO
         ORG,1     O'SSSS'
         SYSTEM    PDP11
            .
            .
            .
           REMAINDER OF PROGRAM DECK
            .
            .
            .
         END    STARTING ADDRESS
!LOAD (BIAS,0),(MAP),(NOSYSLIB),(NOTCB),(ABS),(LMN,PDPL),(GO)
!ASSIGN M:PO,(DEVICE,PPA01),(OUT)
!ASSIGN M:EI,(FILE,PDPL)
!RUN (LMN,SLOAD)
!FIN
```

V. Progress

SAPDP has been successfully used to assemble on the Sigma 5 computer and create an executable object tape of a PDP-11 debug-type program which can, under teletype request, display on the teletype or change the contents of any memory cell. While this small program did not exercise METASYMBOL to any great extent, it has shown that the convenience features of Sigma MACRO processing and high-speed input/output have been made available for PDP-11 programming. Future planned use of SAPDP includes the programming of the software portion of the μ-2 Fast Acquisition Ranging System on a soon-to-be-delivered PDP-11.

Table 1. Address mode syntax

Address mode number	Type of addressing	PAL-11	SAPDP
0R	Register	R	RE
2R	Auto increment	(ER)+	(@I,RE) or (RE,@)
4R	Auto decrement	−(ER)	(@D,RE) or (@,RE)
6R	Index	E(ER)	(E,RE)
27	Immediate	#E	(#,E)
67	Relative address	E	E
37	Absolute address	@#E	*(#,E)