January 1967

The PDPMAP Assembly System

Thomas H. Johnson
University of Pennsylvania

Michael S. Wolfberg
University of Pennsylvania

Follow this and additional works at: https://repository.upenn.edu/cis_reports

Recommended Citation


This paper is posted at ScholarlyCommons. https://repository.upenn.edu/cis_reports/794
For more information, please contact repository@pobox.upenn.edu.
The PDPMAP Assembly System

Abstract
This report describes the PDPMAP Assembly System which is used to assemble symbolic programs written for a Digital Equipment Corporation PDP-8 or DEC-338 with up to 16-K memory locations. The system is implemented at the University of Pennsylvania on an IBM 7040 and DEC PDP-8 connected by a high-speed data channel (IBM 7904 and DEC DM03). The PDPMAP System uses the powerful assembler of a larger computer (IBM 7040 MAP Assembler) to quickly assemble programs for a small computer.

Comments

This technical report is available at ScholarlyCommons: https://repository.upenn.edu/cis_reports/794
INTERIM TECHNICAL REPORT

THE PDPMAP ASSEMBLY SYSTEM

by

Thomas H. Johnson
Michael S. Wolfberg

The following report was prepared under
Contract NOnr 551(40)
for the
Office of Naval Research
and
Rome Air Development Center
and under
Contract AF 49(638)-1421
for the
Air Force Office of Scientific Research

10 October 1967

Moore School Report No. 68-11
Distribution of This Document is Unlimited
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Outline of the Assembly System</td>
<td>1</td>
</tr>
<tr>
<td>3. Outline of the Assembly Process</td>
<td>3</td>
</tr>
<tr>
<td>4. Introduction to the Language</td>
<td>4</td>
</tr>
<tr>
<td>5. Absolute vs. Relocatable Assembly</td>
<td>5</td>
</tr>
<tr>
<td>6. PDP-8 Instruction Format</td>
<td>6</td>
</tr>
<tr>
<td>7. DEC-338 Instruction and Command Format</td>
<td>7</td>
</tr>
<tr>
<td>8. Data Generation and Literals</td>
<td>8</td>
</tr>
<tr>
<td>9. Origins, Field Setting</td>
<td>9</td>
</tr>
<tr>
<td>10. Location Counters</td>
<td>11</td>
</tr>
<tr>
<td>11. Location Symbol Definitions</td>
<td>12</td>
</tr>
<tr>
<td>12. Absolute Symbol Definitions</td>
<td>13</td>
</tr>
<tr>
<td>13. Operation Code Definitions</td>
<td>13</td>
</tr>
<tr>
<td>14. Address and Data Transformation</td>
<td>14</td>
</tr>
<tr>
<td>15. Memory Reference Instructions and Linking</td>
<td>14</td>
</tr>
<tr>
<td>16. Off-Page Linking</td>
<td>15</td>
</tr>
<tr>
<td>17. SYMTAB</td>
<td>17</td>
</tr>
<tr>
<td>18. Operation of the 7040</td>
<td>18</td>
</tr>
<tr>
<td>18.1 TEST</td>
<td>19</td>
</tr>
<tr>
<td>18.2 PUNCH</td>
<td>21</td>
</tr>
<tr>
<td>18.3 Loading the PDP-8</td>
<td>26</td>
</tr>
<tr>
<td>19. Operation of the PDP-8</td>
<td>27</td>
</tr>
<tr>
<td>19.1 PDPMAP-PUNCH</td>
<td>27</td>
</tr>
<tr>
<td>19.2 PDPMAP-LOADER</td>
<td>29</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>20. Postprocessor Output</td>
<td>30</td>
</tr>
<tr>
<td>21. PDPMAP Definitions</td>
<td>32</td>
</tr>
<tr>
<td>21.1 PDP-8 PDPMAP Definitions</td>
<td>32</td>
</tr>
<tr>
<td>21.2 DEC-338 PDPMAP Definitions</td>
<td>33</td>
</tr>
<tr>
<td>22. Defined Operations and Commands</td>
<td>34</td>
</tr>
<tr>
<td>22.1 Initially Defined Operations for PDP-8 PDPMAP</td>
<td>34</td>
</tr>
<tr>
<td>22.2 Initially Defined Operations for DEC-338 PDPMAP</td>
<td>35</td>
</tr>
<tr>
<td>22.3 DEC-330 Control State Commands (microprogrammable)</td>
<td>36</td>
</tr>
<tr>
<td>23. Sample Assembly</td>
<td>37</td>
</tr>
<tr>
<td>23.1 Listing</td>
<td>37</td>
</tr>
<tr>
<td>23.2 Explanation</td>
<td>44</td>
</tr>
<tr>
<td>References</td>
<td></td>
</tr>
</tbody>
</table>
THE PDPMAP ASSEMBLY SYSTEM

1. Introduction

This report describes the PDPMAP Assembly System which is used to assemble symbolic programs written for a Digital Equipment Corporation PDP-8 or DEC-338 with up to 16-K memory locations. The system is implemented at the University of Pennsylvania on an IBM 7040 and DEC PDP-8 connected by a high-speed data channel (IBM 7904 and DEC DM03). The PDPMAP System uses the powerful assembler of a larger computer (IBM 7040 MAP Assembler) to quickly assemble programs for a small computer.

2. Outline of the Assembly System

Motivation for construction of the PDPMAP Assembly System stemmed from the problems involved with assembly on the PDP-8 using the standard PAL or MACRO-8 Assemblers. Great amounts of time were used for input/output on the ASR-33 Teletype. As users began writing long programs, they were faced with assembly times measured in hours. Furthermore, the limitation on the number of symbols which could be defined using these assemblers was a severe handicap to programmers.

A PDP-8 assembly system using the 7040 was written at the Moore School by Paul R. Weinberg and Michael S. Wolfberg. This system was satisfactory for a PDP-8 with 4-K memory, but was inadequate for any larger memory or for the DEC-338. Furthermore, in using the assembly system a great amount of programming effort was required to do off-page linking or to assemble complemented variables. By rewriting the system, a more powerful assembly system was produced which provided for the equipment currently in use and for the possible expansion of the equipment, as well as relieving the programmer of some of the problems encountered using the previous assembly system.
The MAP Assembler (IBMAP) is a sophisticated macro assembly program included in the IBSYS operating system for the IBM 7040. IBMAP is used to perform the initial assembly in PDPMAP, and the remaining tasks are carried out by a postprocessor in the 7040. By using the OPD (operation code definition) pseudo-operation of MAP, PDP-8 and DEC-338 operation codes are defined so they are recognizable as MAP instruction mnemonics. Although macros could have been used, the chosen method is significantly faster and produces a more readable assembly listing.

The postprocessor transforms the assembled code and packs it into 12-bit code for the PDP-8. The postprocessor can create off-page links under programmer control and performs other minor functions to enhance programming ease.

The major features of the PDPMAP System are:
1. Fast assembly (a 4-K core load can be assembled on the 7040 in roughly 4 minutes).
2. Similarity with PAL and MACRO-8 and using card input of fixed format.
3. Complete macro capabilities of MAP.
4. Literals (somewhat more limited than the literal facility of MACRO-8).
5. Virtually unlimited symbol table.
6. Automatic creation of off-page links (greater capability than with MACRO-8).
7. Multiple location counters.
8. Line printer listings.
3. **Outline of the Assembly Process**

The PDPMAP System's assembly process is a multifunctional operation. Different outputs and combinations of outputs can be produced by the Assembly System (see Sections 18 and 19). The assembly process varies mainly in its output. The illustration of process flow discussed below is a general description even though it deals with specific outputs.

A 7040 card deck is submitted as a standard job to be batch processed. The deck consists of

1) IBSYS control cards,
2) the assembly source deck, and
3) a 7040 object program - the PDPMAP Postprocessor - in relocatable binary form.

The MAP Assembler is loaded in the 7040 under directions of IBSYS control cards. The assembly source deck consists of the symbolic program and assembly definitions which must appear at the start of the assembly source deck to condition the MAP Assembler so that it will create a 36-bit "intermediate code" for every assembled line of source code. The intermediate code includes full 15-bit addresses separate from operation codes and indirect bits. Memory reference instructions are tagged so that they can later be easily identified. During the assembly process, a table of used PDP-8 memory consisting of pairs of memory locations, and a table of available linking areas consisting of starting word addresses is maintained.

After the intermediate code has been created (i.e., assembled) by IBMAP, it and the PDPMAP Postprocessor are loaded by the IBSYS loader. Control in the 7040 is given to the PDPMAP Postprocessor.
The PDPMAP Postprocessor scans the intermediate code, word by word, and converts the 36-bit code to the 12-bit code of the PDP-8. If the word being scanned represents a memory reference instruction, the page bit is determined, and it, the indirect bit, and the 7-bit page address are merged with the operation code to form the instruction. If the word being scanned represents an off-page memory reference instruction, a link word containing the off-page address is created on the current page or the initial page of the current field, and a statement of the creation of a link is printed. The off-page memory reference instruction is then changed to a memory reference instruction directly addressing the link word, and the indirect bit is added.

Messages printed by the postprocessor indicate creation of links, illegal off-page references, and insufficient room for creating links.

Prior to this point in the process the PDP-8 is loaded with a short program called PDPMAP-PUNCH in the form of a binary punched paper tape. When PDPMAP-PUNCH is started, it enters a wait loop for the 7040.

Having completed the conversion of intermediate code to 12-bit code, the postprocessor sends the code to the PDPMAP-PUNCH program waiting in the PDP-8. This code is packed so large assemblies can be stored in a small area. PDPMAP-PUNCH performs a checksum of the 12-bit code, dismisses the 7040, and punches the assembled program in the University of Pennsylvania Compressed Binary Loader 8-channel format. The symbol table can be optionally punched in the CBL format for later use with DDT-UP Octal-Symbolic Debugging Program.

4. Introduction to the Language

A set of symbolic cards, PDPMAP Definitions, conditions the MAP Assembler (IBMAP) to accept the PDPMAP Assembly Language. Since IBMAP performs the assembly, PDPMAP source cards are prepared in the same format
as MAP source cards.

A source deck to be assembled begins with a $IBMMAP card and a set of thirty cards called PDP-8 PDPMAP Definitions. The DEC-338 PDPMAP Definitions for assembly of DEC-338 programs consist of forty-nine cards. The two sets of PDPMAP Definitions are listed in Sections 21.1 and 21.2.

It is highly recommended that the programmers of this language be rather familiar with the MAP Language, especially with macros. A more thorough facility with MAP can be most helpful in effectively writing programs. The following description of PDPMAP Assembly Language assumes some knowledge of the MAP manual.

5. Absolute vs. Relocatable Assembly

The PDPMAP Assembler assembles only absolute programs for the PDP-8, i.e., the exact loading address is specified at assembly time. The implementation of this assembly system on the 7040, however, uses the MAP Assembler to produce a relocatable binary deck which corresponds to the absolute PDP-8 program. This has made it possible to use the MAP listing in a sensible way - the relocatable location 00000 of the 7040 binary deck corresponds to absolute location 00000 of the PDP-8 program.

In PDPMAP, the symbol "LOC.0" (the last character is a zero) names this relocatable location 00000. In cases where absolute quantities must be specified, such as in a subfield of VFD pseudo-operation, it may be necessary to subtract LOC.0. In cases where relocatable quantities must be specified, particularly in the variable field of a memory reference instruction, it may be necessary to add LOC.0.
6. PDP-8 Instruction Format

All PDP-8 operation codes should be punched on cards beginning at column 8. In order to specify indirect addressing for a memory reference instruction, an "*" is included directly following the operation code mnemonic. Memory reference instructions must have an address specified in the variable field. An optional second subfield of the variable field can be used to control off-page linking options. These are discussed in Sect. 16.

Input/Output instructions may stand alone or take one subfield in the variable field. This is useful for CDF and CIF instructions, but the programmer should beware that numbers are normally considered as decimal by the MAP Assembler.

Operate instructions may stand alone or take one subfield in the variable field. Operate microprogramming may be done with two Operate mnemonics by including one in the operation field and one in the variable field. A combination of three or more Operate instructions is specified by including one of them in the operation field and the others in the variable field separated by plus signs (i.e., summed together). When CLA is included with one or more Group 2 Operate instructions, it should be in the variable field and not in the operation field.

There is one exception to the above rules - an excess of 8 must be subtracted from the variable field expression in a combination including SPA, SNA, and SZL.

The PDPMAP Assembler creates a 36-bit intermediate code which for memory reference instructions is:

1) a 1 in bit 2,
2) the indirect flag in bits 12-13,
3) the PDP-8 operation code in bits 15-17,
4) the linking option control in bits 19-20, and
5) the full PDP-8 address in bits 21-35.

The Operate and Input/Output instructions are assembled in the rightmost 12 bits of the 36-bit word. This code appears on the listing printed by the Assembler.

Examples: TAD *+5
DCA* XYZ
AND XYZ,2
KSF
CDF 8
CLL
STL RTR
SZA HLT
CLA CLL+CMA
SMA SZA+CLA
SPA SNA+SZL-8

7. DEC-338 Instruction and Command Format

In addition to the PDP-8 instructions the DEC-338 PDPMAP Definitions define the twenty Input/Output instructions for the PDP-8 to interact with the DEC-330 Display Processor.

Only Control State Commands are defined for the DEC-330 by the DEC-338 PDPMAP Definitions. They are defined as microprogramming mnemonics to be included in either the operation field or the variable field. Display command microprogramming is specified in the same format used for Operate microinstructions (see Sect. 6).
The JUMP and PJUMP commands are defined as macros which must take a relocatable address and may take a second subfield specifying the extra options which can be combined with these commands.

Examples: SC2

```
INT 5
LPON SC1+INT+7
EDS SVEC
POP
SK2 INV+COAT
SCUP INTON+BKOF
JUMP XYZ
PJMP XYZ,LPON+SC2
```

8. Data Generation and Literals

Data words in PDPMAP can be specified by PZE, MZE, DEC, and OCT pseudo-operations and by decimal and octal literals.

If literals are specified in any place in the program either LORG or LITORG pseudo-operations must appear in the PDPMAP source deck.

Examples: PZE 405

```
OCT 77
PZE -8
MZE 8
TAD =-10
AND =07700
PZE LOOP
MZE LOOP+2
LITORG
```
9. Origins, Field Setting

At the beginning of an assembly, a default origin of 00000 (LOC.0) is assumed. In order for the PDPMAP Assembly System to be informed of the limits of assembly, the programmer must use one of four special pseudo-operations in order to specify new origins. Each of these pseudo-operations is actually a MAP macro which includes entering the pair of limits into a table of pairs. The // location counter is used to keep track of the pair table. The origin macros depend upon created symbols being ON, which is normally the situation unless a programmer uses the NOCRS pseudo-operation.

Because of the implementation, using BSS or BES pseudo-operations cause the assembly of words (usually zeros) for the PDP-8 in addition to allocating space. It is therefore recommended that origin pseudo-operations be used to allocate blocks of storage.

The following four pseudo-operations are available for specifying origins:

**OCTORG**

This pseudo-operation sets the current assembly location counter to the address specified in the variable field. The address must be an octal number between 0 and 37777.

Examples: OCTORG 10

OCTORG 10000

**ABSORG**

This pseudo-operation sets the current assembly location counter to the address determined by the expression in the variable field. The expression may be any absolute expression whose value is between 0 and 16383.
Examples:  ABSORG 8
ABSORG 4096
(Note:  These uses of ABSORG are equivalent to the uses of OCTORG in the examples above.)
ABSORG LOOP-LOC.0
ABSORG SIZE*SIZE+18
(where LOOP is a relocatable symbol and SIZE is an absolute symbol)

RELORG

This pseudo-operation sets the current assembly location counter to the address determined by the expression in the variable field. The expression may be any relocatable expression whose value is between relocatable 0 and 16383.

Examples:  RELORG LOOP+15
RELORG LOC.O+SIZE*SIZE+18
RELORG *+10
(where LOOP is a relocatable symbol and SIZE is an absolute symbol)

PAGE

This pseudo-operation sets the current assembly location counter to the beginning of a PDP-8 page according to the variable field.

a) A decimal page number between 0 and 127 can be given which specifies a particular page origin within the 16-K memory.

b) When the variable field is left blank (i.e., no argument is given), the current assembly location counter is set to the beginning of the next PDP-8 page. If the location counter is already at the beginning of a page, this pseudo-operation has no effect. When no argument is given, a severity zero error message is generated.
and should be ignored.

Examples: PAGE 1

    PAGE

    PAGE 38

FIELD

This pseudo-operation is used to specify the memory field implicitly used by the following pseudo-operations: OCTADR, OCTORG, ABSORG, PAGE (with an argument). It is not necessary to use this pseudo-operation to assemble for higher memory fields; it is included in the PDPMAP System to make it an easy task to take a program previously assembled for one memory field and assemble it for another.

The FIELD pseudo-operation sets the implicit field specification to the set value of the expression in the variable field. This value should be between 0 and 3. In further uses of OCTADR, OCTORG, ABSORG, or PAGE (with an argument), 4096 times the implicit field number is added to the value of the specified expression.

Examples: FIELD 1

    FIELD 0

    FIELD A

10. Location Counters

The PDPMAP Assembly System uses the // location counter to keep track of limits of assembled words, and therefore the programmer is not permitted to use it. The programmer may, however, define location counters and use them along with the blank location counter if he does so through a special macro. The programmer must not use the USE pseudo-operation.
If a programmer wishes to use location counters in a program, he should include the following macro definition at the beginning of his source deck:

```
USE. MACRO X,Y,Z
  Z RELORG Y
  ORG Z
  USE X
  Y BSS 0
ENDM USE.
```

This macro can be called in the same way that the USE pseudo-operation is normally called except "USE. PREVIOUS" is not allowed and the specification to use the blank location counter should have "( )" in the variable field.

The EVEN pseudo-operation of MAP may not be used in PDPMAP.

11. Location Symbol Definitions

A location symbol must be relocatable in PDPMAP. As usual, a location symbol may be defined by including it in the location field of a card containing PDPMAP assembly code in the operation and variable fields. The EQU pseudo-operation can also be used to define a location symbol as long as the variable field is a relocatable expression.

A special pseudo-operation is available in PDPMAP for the definition of a location symbol as an address specified by an octal number.

```
OCTADR
```

This pseudo-operation takes two arguments - a symbol and an octal number (between 0 and 37777) separated by a comma. Its effect is to define the first argument as a relocatable symbol corresponding to the second argument.
Examples: OCTADR XRL,10010
OCTADR START,200

The programmer should beware not to define location symbols which correspond to any microprogramming mnemonics because these symbols are defined by SET pseudo-operations for use in the variable field of PDP-8 of Operate instructions or DEO-330 commands. Also, LOC.0 is a reserved location symbol naming PDP-8 location 00000, and FLD. is a reserved set symbol used to keep track of the implicit field as set up by the FIELD pseudo-operation.

12. Absolute Symbol Definitions

The EQU, BOOL, and SET pseudo-operations may be used to define absolute symbols. As mentioned in the above section, the programmer should be careful not to define any symbols which correspond to microprogramming mnemonics.

13. Operation Code Definitions

The operation code "END." is used by the PDPMAP Assembly System and should not be redefined by the programmer.

The programmer may define synonyms for any of the six memory reference instruction mnemonics by the OPSYN pseudo-operation.

A special pseudo-operation is provided for the definition of new Input/Output (IOT) and Operate (OPR) instruction mnemonics.

**DEFINE**

This pseudo-operation takes two arguments - a symbol and an octal number (less than 10000), separated by a comma. Its effect is to define the first argument as an operation code whose value is the second argument. The operation code may either be used alone or with one subfield in the variable field. If the second argument is greater than or equal to 7000
(octal), the first argument is also defined with a SET value corresponding to the low-order eight bits of the second argument. This makes it possible to include the symbol in the variable field of an Operate instruction for microprogramming purposes.

Examples:  DEFINE STA,7240
            DEFINE LCC,6703

14. Address and Data Transformation

The PDPMAP Postprocessor transforms the 36-bit intermediate code assembled by MAP into 12-bit PDP-8 code. The absolute value of absolute expressions to be assembled must be less than 4096 (10000 octal) or greater than 28672 (70000 octal), otherwise the quantity is assumed to be an address which must be unrelocated (i.e., the postprocessor will subtract LOC.0 from the address). For all absolute expressions the low order 12 bits of the address are used. This makes it possible to specify expressions with a negative value in the address of a PZE.

If the sign bit of the 36-bit intermediate code is a ONE the two's complement of previously computed 12-bit quantity is taken. This gives the programmer the power to specify negative numbers in literals or in OCT or DEC pseudo-operations. It also permits the programmer to specify the complement of a relocatable address by using MZE. Using PZE with a negative address does not work for this purpose.

15. Memory Reference Instructions and Linking

The PDPMAP Postprocessor recognizes a memory reference instruction when bit 2 of the 36-bit word is a ONE. If the instruction addresses a location on the same page or initial page of the current field, a 12-bit PDP-8 word is computed. If the memory reference instruction does not have an indirect
flag and if the address cannot be directly addressed, the postprocessor will attempt to create a link. The linking algorithm is described below.

If the memory reference instruction has an indirect flag and if the address cannot be directly addressed in the PDP-8, an error diagnostic is listed and the 12-bit quantity is computed as if the address were on the same page. If the memory reference instruction has an indirect flag and if the address is on a different field than the current location, an error diagnostic is listed and the 12-bit quantity is computed as if the address were on the same field using the previous algorithm for indirect addressing.

16. Off-Page Linking

The programmer using PDPMAP does not have to perform the bookkeeping often needed to properly address locations in the PDP-8. He can let the postprocessor create the necessary links to referenced locations when necessary. The programmer can control the postprocessor's linking through a special pseudo-operation and some options to alter the normal linking algorithm.

LINKS

This pseudo-operation is used to indicate where on a page links can be used or created. The use of this pseudo-operation does not alter the current location counter; it indicates that the locations between its occurrence and the end of the page or the next origin in the source deck, are considered available linking area. If the next origin is beyond the end of the page the area stops at the end of the page. If the next origin is less than the current location the link area goes to the end of the page.

The LINKS pseudo-operation can be used any number of times.

Any unused location in the available linking area is considered a possible location for the generation of a link word. Any used location in
the available linking area which does not contain a memory reference instruction is considered an existing usable link word according to the 12-bit quantity assembled there.

In the absence of any programmer alteration, links are created for memory reference instructions which reference locations not on the current page or the initial page. Exceptions occur with references to locations on other fields. For a direct JMP or JMS instruction referencing a location on a field other than the current one, the postprocessor assumes a CIF instruction has been given previously and will assemble the instruction as if the referenced location were in the current field. For a direct AND, TAD, ISZ or DCA instruction referencing a location on a field other than the current one, the postprocessor assumes a CDF instruction has been given previously and assembles the instruction with a link no matter what location is being referenced.

The linking algorithm used by the postprocessor in the absence of any programmer alteration options is:

1. Attempt to use an existing linking word on current page.
2. Attempt to use an existing linking word on the initial page of the current field.
3. Attempt to create a linking word on the current page.
4. Attempt to create a linking word on the initial page of the current field.
5. If all attempts fail, the 12-bit quantity is computed as if the address were on the same page except cross-field references to an initial page are computed as if the address were on the initial page of the current field.
6. If all attempts fail (1-4) an error diagnostic is listed.
The PDPMAP Postprocessor operates on the coding in order, from location 0 through the end of memory. Links are created in the order they are processed and not the order in which the program was assembled. In a similar fashion, the linking areas specified by the LINKS pseudo-operation are filled from low address to high address. If there is more than one linking area on a page the one with the lower address is filled first.

It is important to note that by proper use of the origin pseudo-operations the programmer can overlay part of his coding. This does not overlay the LINKS pseudo-operation. The linking area specified early in an assembly will remain a linking area even though new code may overlay it.

Extra control over the linking algorithm is provided by using the tag (second subfield of the variable field). The following three options are provided:

<table>
<thead>
<tr>
<th>TAG</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Do not link - if a link is needed, assemble as if the address is on the current page except cross-field references to an initial page, then assemble as if the address were on the initial page of the current field (Step 5 only)</td>
</tr>
<tr>
<td>2</td>
<td>Consider linking on the initial page only (Steps 1, 3, 5, and 6)</td>
</tr>
<tr>
<td>3</td>
<td>Consider linking on current page only (Steps 2, 4, 5, and 6).</td>
</tr>
</tbody>
</table>

Any tags greater than 3 are ignored. When no linking is needed, the tag is ignored.

17. SYMTAB

It is often desirable to obtain a symbol table paper tape as an auxiliary output from the PDPMAP assembly process. This tape can be later read into DDT-UP so the user may debug his program using the same symbols in which his program was written. Such a symbol table includes only location symbols and those defined by EQU, BOOL, and OCTADR pseudo-operations. No created symbols or operation codes are included in the symbol table.
The creation of the symbol table is performed on the 7040 by the SYMTAB program, and the results are directly sent to the PDP-8 to be punched as a CBL binary paper tape by the PDPMAP-PUNCH program. SYMTAB reads a Debug Dictionary produced by IBMAP as input data to get the information necessary to output a DDT-UP symbol table. The Debug Dictionary is normally obtained in a 7040 IBSYS/IBJOS run to facilitate the use of an execution-time debugging package. It is punched by IBMAP when a binary deck is punched and when the DD option is included on the $IBMAP card.

In the PDPMAP System, a user may obtain his symbol table tape when the assembly is performed or it may be obtained by a separate run. The various ways of setting up the 7040 run are individually discussed below. Operation of the PDPMAP-PUNCH program used to punch a symbol table on the PDP-8 is described in Section 19.1.

The SYMTAB program has the option, according to which entry point is used, of punching any binary deck it reads (except for the Debug Dictionary). This is useful in the type of run described in Section 18.2.

18. Operation of the 7040

The 7040 jobs which are used in the PDPMAP System execute under standard batch IBSYS. This section describes the ways in which these jobs are prepared. There are three basic options of the PDPMAP Postprocessor: TEST, PUNCH, LOAD. The option is indicated by punching one of the three in column 16 of the PDPMAP Postprocessor's $ENTRY card. There is also the facility to obtain a DDT-UP symbol table, and there is the facility to punch a binary relocatable card deck.

The most common variations of all these options are described below, including listings of control cards to be used. In the listings of control cards, the symbol "NAME" is the arbitrarily chosen name of the user's PDP-8 program.
In any of the variations which include the PDP-8, either machine may begin execution before the other.

18.1 TEST

The Testing processes take place entirely in the 7040 with no interaction with the PDP-8. Testing is useful for checking the assembly or determining if linking can be performed when the PDP-8 is not available or when using a 7040 with no PDP-8.

Two levels of testing are available.

Assemble Only

The first level of testing only uses the MAP assembler to assemble the user's symbolic deck. This type of run catches undefined and multiply defined symbols, illegal operation codes, problems in macro expansions, etc. This run is set up as follows:

$JOB ...
$IBJOB
$IBMAP NAME

PDFMAP
definitions

User's
program

END

$IBSYS

Assemble and Test

A second level of testing may be performed by running the PDPMAP Postprocessor on the 7040 to create links and check for illegal addressing. This level of testing will be performed as long as a binary deck would normally be punched by MAP. Low severity errors such as undefined and multiply defined symbols will not stop execution of the PDPMAP Postprocessor. The occurrence
of an illegal operation code, however, will block the second level of testing, but the first level of testing is still performed. This use of the PDPMAP Postprocessor on the 7040 does everything the program does when the 'PUNCH' option is used except results are not sent to the PDP-8 (Section 18.2). Therefore, this testing could be performed on another 7040.

The run is set up as follows:

```
$JOB ...
$RESTORE
$CLOSE S.SU15,REWIND
$SWITCH S.SPP1,S.SU15
$IBJOB ' DECK
$IBMAP NAME

    PDPMAP definitions

    User's program

    END

$IBSYS
$SWITCH S.SPP1,S.SU15
$CLOSE S.SU15,MARK,REWIND
$IBJOB PDPMAP NOMAP
$IBLDR PDPMAP

    Postprocessor binary deck

$END PDPMAP
$EDIT U15
$IBLDR NAME
$EDIT IN
$ENTRY TEST
$IBSYS
$RESTORE
$IBSYS

Note that the name of the user's program on the $IBMAP card must correspond to the name on the last $IBLDR card.
Assemble, Get Deck, Test

Since the 7040 connected to the PDP-8 uses an on-line card reader
and line printer, it may be desirable to run a long assembly on another 7040
and just punch a binary deck. In addition to punching, the user may as
well test the results by executing the PDPMAP Postprocessor in the test mode.
Such a run is set up as follows:

$JOB ...
$1BJOB PDPMAP NOMAP,DECK
$1BLDR PDPMAP
   Postprocessor
   binary deck
$1KEND PDPMAP
$1BMAP NAME
   PDPMAP
   definitions
   User's
   program
   END
$ENTRY TEST
$IBSYS

18.2 PUNCH

If the user wishes to obtain a CBL tape of his program, the PDPMAP
Postprocessor must be run on the 7040 and the PDP-8 must be loaded and started
with the PDPMAP-PUNCH program. The operation of the PDPMAP-PUNCH program on
the PDP-8 is described in Section 19.1. The 'PUNCH' option is the standard
PDPMAP option with the variation of runs listed below.

Assemble, Get Tape

The recommended standard PDPMAP run includes both the assembly of the
symbolic deck and the execution of the PDPMAP Postprocessor. No binary cards
are punched by this type of run. The run is set up as follows:
Note that the only difference between the above set of cards and those in Section 18.1 is with the $ENTRY card. Alternatively, the $ENTRY could be included with no name. In this case the assumed option is 'PUNCH'.

Assemble, Get Tape - Method No. 2

If the user is very sure of his program's correctly assembling, he may wish to use a set of control cards simpler than those listed above to achieve the same effect. The run is set up as follows:
Assemble, Get Deck, Get Tape

If the user wishes to assemble, obtain a binary deck, and punch a CBL tape all in one run, then he must have an assembly with no errors of severity greater than one. Undefined or multiply defined symbols will block the execution of the PDPMAP Postprocessor although a binary deck will still be punched. It could subsequently be included in a run of the type described in the next subsection to create the CBL tape.

A run of this type is set up as follows:

```
$JOB ...
$IBJOB PDPMAP NOMAP
$IBLDR PDPMAP
    Postprocessor
    binary deck
$DKEND PDPMAP
$IBMAP NAME
    PDPMAP
    definitions
    User's
    program
    END
$ENTRY PUNCH
$IBSYS
```

```
$JB$JOB ...
$IBJOB PDPMAP NOMAP,DECK
$IBLDR PDPMAP
    Postprocessor
    binary deck
$DKEND PDPMAP
$IBMAP NAME
    PDPMAP
    definitions
    User's
    program
    END
$ENTRY PUNCH
$IBSYS
```
Get Tape

If the user has already obtained a binary deck (such as in the previous subsection), then he may submit a 7040 run which will be very quick. This run will result in a CBL tape in addition to the output listing generated by the PDPMAP Postprocessor on the 7040.

The run is set up as follows:

```
$JOB ...
$IBJOB PDPMAP NOMAP,NOSOURCE
$IBLDR PDPMAP
        Postprocessor
        binary deck
$DKEND PDPMAP
$IBLDR NAME
        binary deck
$DKEND NAME
$ENTRY PUNCH
$IBSYS
```

Assemble, (Get Binary Deck), Get CBL Tape, Get Symbol Table

This variation is the recommended standard for short or moderate length assemblies. When running this variation, the PDP-8 should be set up to automatically restart the PDPMAP-PUNCH program (see Section 19.1).

The user has the option of obtaining a binary deck (without the Debug Dictionary). The $ENTRY card for SYMTAB is used to indicate whether a binary deck is punched. The user may specify either 'DECK' or 'NODECK' starting in column 16, or if neither is there the assumed option is NODECK.

The run is set up as follows:
Note that the name of the user's program on the $IBMAP card must correspond to the name on the last $IBLDR card.
Symbol Table Only

If the user previously assembled his program and obtained a binary deck including a Debug Dictionary, then he may wish to perform this type of run. Such a binary deck could have been punched as described in Sections 18.1 or 18.2 except, in addition, the options on the $IBMAP card should include 'DD'.

The only data cards necessary for this job are the Debug Dictionary cards from the $DDICT through the last binary card before $TEXT. The user may, however, include the entire binary deck as data, and the extra cards will be ignored.

SYMTAB will read any number of Debug Dictionaries provided as input so that symbol tables of separate assemblies could be combined at this level of operation since one DDT-UP symbol table tape results from this process.

The run is set up as follows:

$JOB ...
$IBJOB SYMTAB NOSOURCE,NOMAP
$IBLDR SYMTAB

binary deck
$DKEND SYMTAB
$ENTRY
$DDICT NAME

binary cards
Debug Dictionary
$IBSYS

18.3 Loading the PDP-8

A program which has been assembled by the PDPMAP Assembler may be quickly loaded into the PDP-8 directly from the 7040. The binary deck of the assembled program is used, and the LOAD option of the PDPMAP Postprocessor is used (by specifying 'LOAD' on the $ENTRY card). This method of loading always
loads PDP-8 locations 0000 through 7575, and locations where nothing was assembled are cleared to 0000.

Operation of the PDPMAP-LOADER program used on the PDP-8 is described in Section 19.2.

This run is set up as follows:

$JOB ...
$IBJOB PDPMAP NOSOURCE,NOMAP
$IBLDR PDPMAP
  Postprocessor
  binary deck
$IDKEND PDPMAP
$IBLDR NAME
  binary deck
$IDKEND NAME
$ENTRY LOAD
$IBSYS

19. Operation of the PDP-8

This section describes the operations of the PDP-8 in the PDPMAP Assembly System. There are two PDP-8 programs involved. The PDPMAP-PUNCH program is used to receive data from the 7040 and punch a CBL binary tape (either user's program or symbol table). The PDPMAP-LOADER program is used to receive data from the 7040 such that the PDP-8 is quickly loaded, and there is an option to immediately begin execution.

19.1 PDPMAP-PUNCH

This program is loaded into locations 0000 through 0377 of the PDP-8 by the CBL Loader. The operation of this program follows:

a) START the PDP-8 at location 0000. The program goes into a loop to wait for the 7040 to send the data. If the 7040 was already waiting, transmission occurs immediately.
b) If no successful transmission was made, the program remains in the wait loop for the 7040. If transmission was completed, a checksum of the data is computed. If the checksum is correct, the program proceeds to examine the SWITCH REGISTER setting. If the checksum is incorrect, the transmission is attempted nine more times. Should all ten attempts fail, the PDPMAP Postprocessor prints an error message and returns control to IBSYS.

c) If bit 0 of the SWITCH REGISTER is a ONE, the PDP-8 halts after successful transmission. Depressing CONTINUE causes the program to proceed to the next step.

d) The CBL tape is punched beginning with a few inches of leader and ending with a few inches of blank trailer. Be sure the Teletype punch is turned on and the Teletype is on-line.

e) After the tape has been punched, the SWITCH REGISTER is again examined by the program. If bit 1 of the SWITCH REGISTER is a ZERO, the PDP-8 halts. Depressing CONTINUE causes the program to proceed to the next step.

f) If bit 1 of the SWITCH REGISTER is not ZERO, the program is restarted at location 0000 for another job. If the 7040 had already been waiting for the PDP-8, transmission will occur immediately.

The first option of halting (step c) is useful when only one PDPMAP punching is desired. The PDP-8 can be set up to wait (unattended) with its Teletype turned off until the 7040 PDPMAP job is run. If successful transmission occurs, the PDP-8 will halt. At a later time, the user may return to find the job completed. He then may turn on the Teletype to 'LINE', turn on the Teletype punch, and depress CONTINUE to commence punching.
If there is a transmission error, and the user decides to take his chances of punching a possibly invalid tape, he may force the PDPMAP-PUNCH program to begin operation at step d by STARTing at location 0001. This option can also be used to punch more than one copy of the CBL tape or to restart punching if the user forgot to turn on the Teletype punch.

19.2 PDPMAP-LOADER

This program is loaded into locations 7600 through 7667 of the PDP-8 by the CBL Loader. The operation of this program follows:

a) LOAD ADDRESS with 7600.

b) For immediate execution of the program being loaded, set the SWITCH REGISTER to the appropriate starting address. If 7600 is left in the SWITCH REGISTER, the PDP-8 will halt after successful loading.

c) START the PDP-8. The PDPMAP-LOADER goes into a loop to wait for the 7040 to send the data. While it is waiting, the accumulator lights indicate the contents of the SWITCH REGISTER when the program was started. If the 7040 was already waiting, transmission occurs immediately.

d) A checksum of the data is computed, and if transmission was successful, the program acts according to the way it was started in steps b and c. If the halting option was used, depressing CONTINUE causes the PDP-8 to jump to location 7777, which is the starting address of the CBL Loader.

e) If no successful transmission was made, the program remains in the wait loop for the 7040.

The PDPMAP-LOADER is self-initializing and self-preserving. It may therefore be used any number of times.
20. **Postprocessor Output**

When the PDPMAP Postprocessor is operated in 'TEST' or 'PUNCH' mode, a listing is printed indicating the formation of links and errors that occur during postprocessing. In Section 23 there is a sample listing of a PDPMAP program with the postprocessor output at the very end.

The column at the far left of the output labeled PAGE STARTING ADDRESS contains the octal starting addresses of pages assembled by PDPMAP. This address is printed immediately following the other information about the page of assembled code.

The next three columns are utilized when links are created. The column labeled LINKED INSTRUCTION ADDRESS contains the location of an instruction which was linked through the address found in the next column labeled LINK WORD ADDRESS. The value stored in the link word is printed in the next column under LINK WORD VALUE. If this number is in parentheses then the value existed previously.

The last three columns are printed on the same row as the page starting address and they summarize the number of linking operations done on that page. The first two columns, below LINKS CREATED, indicate the number of links created on the current page and the initial page respectively. The last column indicates the number of link words on the current page that are not used by the assembled program.

At the end of every field where linking could be done through its initial page the final status of the initial page is printed out listing the total number of links created on the initial page and the number of link words still available.
When the PDFMAP Postprocessor operates in 'LOAD' mode, only error diagnostics are printed.

**Error Diagnostics**

Error diagnostics at assembly time originate from the MAP assembler. Errors detected in the PDFMAP Postprocessor are concerned with incorrect addressing or communication with the PDP-8.

**UNABLE TO LINK LOC. = [a] TO LOC. = [b]**

There are an insufficient number of link words to link the instruction in location "a" to the address "b". The cause is neglecting to use the LINKS pseudo-operations or running out of link words.

**LOCATION [a] IS INCORRECTLY ADDRESSED TO [b]**

A memory reference instruction with an indirect flag cannot directly address the referenced location. The location is off the page or off the field.

**PROGRAM EXCEEDS PUNCH BUFFER [a] IS THE LAST LOC. PUNCHED**

Only locations from the beginning of core to "a" could be punched at the PDP-8.

**COMMUNICATION ERROR**

The data channel between the 7040 and the PDP-8 has malfunctioned. This error should be brought to the attention of the proper personnel for correction.
21. PDFMAP Definitions
21.1 PDP-8 PDFMAP Definitions

UNLIST PDP-8 PDFMAP DEFINITIONS 7/1/67 (30 CARDS) 00000001
A MACRO H,J,K,L,M,N,P THESE 30 CARDS DEFINE THE BASIC PDP-8 00000002
J INSTRUCTION SET PLUS THE FOLLOWING ...
K L BY MEMORY FIELD SETTING - 'FIELD'
M N M.S. NEW OPR OR IOT DEF'N. - 'DEFINE'
IRP P WOLFBERG OCTAL ADDRESS DEF'N. - 'OCTADR'
P ORIGINS - 'ABSORG', 'RELORG', 'OCTORG', 'PAGE' 00000007
ENDM A REGION FOR OFF-PAGE LINKS - 'LINKS'

LOC.O A END.(OPSYN END)(MACRO X) Defined XENDM(ENTRY LUC.O) 00000009
A END(MACRO X)(BSS O)(USE //)(PZE -1,X)(MC ON)END.MEND) 00000010
A LINKS(MACRO X)(BSS O)(USE //)(MZE X)(USE PREVIOUS)ENDM 00000011
A PAGE(MACRO X,Y)(EQU -(LOC.O-FLD.+127))IFT X 00000012
ETC ((IFF 1\X)(DUP 2,0)(ABSORG Y/128*128)IFF(ABSORG 128*X)ENDM) 00000013
A OCTORG(MACRO X,Y)(BOOL X)(ABSORG Y)ENDM(BEGIN //,**+16384) 00000014
A DEFINE(MACRO X,Y)(OPD 10000*Y*00000000,*,1,2,)ENDM 00000015
A ABSORG(MACRO X,Y,Z)(BSS 0)Z(ORG X+LOC.O+FLD.) 00000016
ETC ((USE //)(PZE Z,,Y)(USE PREVIOUS)ENDM(B ((AND,0)(TAD,1))) 00000017
A RELORG(MACRO X)(ABSORG X-LOC.O-FLD.)ENDM(B ((ISZ,2))) 00000018
A ()(B ((OCA,3)(JMS,4)(JMP,5))FIELD(MACRO X)FLD.(SET X+496)ENDM 00000019
A FLD.(SET 0)(DEF(MACRO X,Y)(OPD Y,,1, (((IF Y/7000)))) 00000020
X SET Y-Y/10*2-Y/100*16-Y/1000*640-Y/10000*256 00000021
ENDM DEFINE
B ((OPR,7000)(NOP,7000)(CLA,7200)(CLL,7100)(CLM,7040)(CML,7020)) 00000023
B ((RAR,7010)(RAL,7004)(RTR,7012)(RTL,7006)(IAC,7001)(SMA,7500)) 00000024
B ((SZA,7440)(SPA,7510)(SNA,7450)(SNL,7420)(SZL,7430)(ORS,7404)) 00000025
B ((HLT,7402)(CIA,7041)(LAS,7604)(SSTL,7120)(GLK,7204)(SKP,7410)) 00000026
B ((IOT,6000)(ION,6001)(IOF,6002)(KSF,6031)(KCC,6032)(KRS,6034)) 00000027
B ((KRB,6036)(TSF,6041)(TGF,6042)(ITPC,6044)(TLS,6046)(CDF,6201)) 00000028
B ((CIF,6202)(RDF,6214)(RIF,6224)(RMF,6244)(RIB,6234)) 00000029
A OCTADR(MACRO X,Y,Z)(BOOL Y)(EQU LOC.O+Z+FLD.)ENDM,LIST END 00000030
Definitions

THESE 49 CARDS DEFINE THE BASIC DEC-338 INSTRUCTION SET PLUS THE FOLLOWING:

1. MEMORY FIELD SETTING - 'FIELD'
2. OCTADR
3. OCTORG
4. PAGE
5. ABSORG
6. RELORG
7. ORIGINS
8. ABSORG
9. PAGE
10. ENOH
11. PAGE
12. ENOH
13. LINKS
14. PAGE
15. DEFINE
16. PAGE
17. PAGE
18. PAGE
19. PAGE
20. PAGE
21. PAGE
22. PAGE
23. PAGE
24. PAGE
25. PAGE
26. PAGE
27. PAGE
28. PAGE
29. PAGE
30. PAGE
31. PAGE
32. PAGE
33. PAGE
34. PAGE
35. PAGE
36. PAGE
37. PAGE
38. PAGE
39. PAGE
40. PAGE
41. PAGE
42. PAGE
43. PAGE
44. PAGE
45. PAGE
46. PAGE
47. PAGE
48. PAGE
49. PAGE
## 22. Defined Operations and Commands

### 22.1 Initially Defined Operations for PDP-8 PDPMAP

<table>
<thead>
<tr>
<th>Memory-Reference Instructions</th>
<th>Group 1 Operate Instructions (microprogrammable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND  0000</td>
<td>OPR  7000</td>
</tr>
<tr>
<td>TAD  1000</td>
<td>NOP  7000</td>
</tr>
<tr>
<td>ISZ  2000</td>
<td>CLA  7200</td>
</tr>
<tr>
<td>DCA  3000</td>
<td>CLL  7100</td>
</tr>
<tr>
<td>JMS  4000</td>
<td>CMA  7040</td>
</tr>
<tr>
<td>JMP  5000</td>
<td>CML  7020</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input/Output Instructions</th>
<th>Group 2 Operate Instructions (microprogrammable)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOT  6000</td>
<td>SMA  7500</td>
</tr>
<tr>
<td>ION  6001</td>
<td>SZA  7440</td>
</tr>
<tr>
<td>IOF  6002</td>
<td>SPA  7510</td>
</tr>
<tr>
<td>KSF  6031</td>
<td>SNA  7450</td>
</tr>
<tr>
<td>KCC  6032</td>
<td>SNL  7420</td>
</tr>
<tr>
<td>KRS  6034</td>
<td>SZL  7430</td>
</tr>
<tr>
<td>KRB  6036</td>
<td>SKP  7410</td>
</tr>
<tr>
<td>TSF  6041</td>
<td>OSR  7404</td>
</tr>
<tr>
<td>TCF  6042</td>
<td>HLT  7402</td>
</tr>
<tr>
<td>TPC  6044</td>
<td>LAS  7604</td>
</tr>
<tr>
<td>TLS  6046</td>
<td></td>
</tr>
</tbody>
</table>

| Extended Memory Instructions |                                              |
|-------------------------------|                                              |
| CDF  6201                     |                                              |
| CIF  6202                     |                                              |
| RDF  6214                     |                                              |
| RIF  6224                     |                                              |
| RMF  6244                     |                                              |
| RIB  6234                     |                                              |
22.2 Initially Defined Operations for DEC-338 PDPMAP

All of the initially defined operations for PDP-8 PDPMAP are included in those for DEC-338 PDPMAP. In addition, the following operations associated with the DEC-338 are defined.

**Input/Output Instructions**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPDP</td>
<td>6051</td>
</tr>
<tr>
<td>RXP</td>
<td>6052</td>
</tr>
<tr>
<td>RYP</td>
<td>6054</td>
</tr>
<tr>
<td>RDAC</td>
<td>6061</td>
</tr>
<tr>
<td>RSI</td>
<td>6062</td>
</tr>
<tr>
<td>RS2</td>
<td>6064</td>
</tr>
<tr>
<td>RPB</td>
<td>6071</td>
</tr>
<tr>
<td>SPDP</td>
<td>6135</td>
</tr>
<tr>
<td>SIC</td>
<td>6145</td>
</tr>
<tr>
<td>LBF</td>
<td>6155</td>
</tr>
<tr>
<td>INIT</td>
<td>6165</td>
</tr>
<tr>
<td>RES1</td>
<td>6174</td>
</tr>
<tr>
<td>RES2</td>
<td>6164</td>
</tr>
<tr>
<td>CFD</td>
<td>6161</td>
</tr>
<tr>
<td>STPD</td>
<td>6154</td>
</tr>
<tr>
<td>SFLP</td>
<td>6132</td>
</tr>
<tr>
<td>SPES</td>
<td>6151</td>
</tr>
<tr>
<td>SPEF</td>
<td>6152</td>
</tr>
<tr>
<td>SPSF</td>
<td>6171</td>
</tr>
<tr>
<td>SPMI</td>
<td>6172</td>
</tr>
</tbody>
</table>
### 22.3 DEC-330 Control State Commands (microprogrammable)

<table>
<thead>
<tr>
<th>Command</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPOF</td>
<td>0040</td>
<td>PEDS</td>
</tr>
<tr>
<td>LPON</td>
<td>0060</td>
<td>PNI</td>
</tr>
<tr>
<td>SC1</td>
<td>0400</td>
<td>PNLS</td>
</tr>
<tr>
<td>SC2</td>
<td>0500</td>
<td>PNM</td>
</tr>
<tr>
<td>SC4</td>
<td>0600</td>
<td>* SK1</td>
</tr>
<tr>
<td>SC8</td>
<td>0700</td>
<td>** INV</td>
</tr>
<tr>
<td>INT</td>
<td>0010</td>
<td>** CLAT</td>
</tr>
<tr>
<td>EDS</td>
<td>1001</td>
<td>** COAT</td>
</tr>
<tr>
<td>CCB</td>
<td>1002</td>
<td>* SK2</td>
</tr>
<tr>
<td>CSB</td>
<td>1004</td>
<td>* SK3</td>
</tr>
<tr>
<td>POINT</td>
<td>1100</td>
<td>* SK4</td>
</tr>
<tr>
<td>INCR</td>
<td>1110</td>
<td>SKIP</td>
</tr>
<tr>
<td>VEC</td>
<td>1120</td>
<td>SNSZ</td>
</tr>
<tr>
<td>VECON</td>
<td>1130</td>
<td>SPBL</td>
</tr>
<tr>
<td>SCVEC</td>
<td>1140</td>
<td>SPB2</td>
</tr>
<tr>
<td>CHAR</td>
<td>1150</td>
<td>SLPSI</td>
</tr>
<tr>
<td>GRAPH</td>
<td>1160</td>
<td>CLPSI</td>
</tr>
<tr>
<td>CLDF</td>
<td>1200</td>
<td>SCUP</td>
</tr>
<tr>
<td>STOP</td>
<td>1400</td>
<td>SCDN</td>
</tr>
<tr>
<td>*** JUMP</td>
<td>2000</td>
<td>INTUP</td>
</tr>
<tr>
<td>*** PJMP</td>
<td>2010</td>
<td>INTDN</td>
</tr>
<tr>
<td>* POP</td>
<td>3000</td>
<td>BKON</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BKOF</td>
</tr>
</tbody>
</table>

* usable only in operation field

** usable only in variable field

*** specially defined to create two words (see Sect. 7)
23. **Sample Assembly**

In order to illustrate the important features of the PDPMAP Assembly System, a DEC-338 program, called "RUBBAN", was written. An annotated assembly listing is provided followed by the PDPMAP Postprocessor output. Also, an explanation of highlights of the assembly and postprocessing is given. The program was specially designed to point out many assembly features, and it is therefore recommended that the reader carefully study the listing and output along with the explanation.

**RUBBAN** is a program which works on a standard DEC-338 with at least 8K memory locations. The program displays a square box on the screen which is used to track the light pen. A vector is drawn from the screen center to the center of the box, giving a rubber band effect as the box is moved around the screen by the user.

The reader who is not interested in DEC-338 programming may ignore the program content, but should still study the assembly features.

### 23.1 Listing

An assembly listing of the program RUBBAN is given here. The listing was produced by IBMAP on the 7040. The first column of the listing indicates PDP-8 locations (in octal) where code was assembled. The next 12 octal digits constitute the intermediate code produced by IBMAP. The next five binary digits are relocation information for the IBSYS loader and may be ignored. Next, the source symbolic card is listed with programmer's comments at the right. The last column of the listing indicates the statement number assigned by IBMAP.
BINDARY CARD CCGCCG

**COMMENT CARD**

- UNLIST DEC-338 PDPMAP DEFINITIONS 7/1/67 GCG000001
- TRACK LIGHT PEN WITH RUBBER-BAND EFFECT
- DEFINE SETONE, 73C1 (CLA, CLL, IAC)
- CCRADRR TEMP1, 600
- CCRADRR TEMP2, 601

**SIDE MACRO**

- SIDE, DELTA TO TRACK WITH SQUARE BOX
- PMC CN
- ELS SVEC DRAW SIDE
- CCT SIDE
- EDS SVEC TEST FOR LIGHT
- EDS SVEC LIGHT SEEN, ADJUST
- GCT DELTA
- PMC OFF
- ENDM SIDE

**OPERATION CODE**

- CCICCC CCG000 C86J4 100001
- CCICCC CCG000 C87000 100001
- CCICCC 1000 15 C CC233 100001

**FULL PDP-8 ADDRESS**

- LINKING AREA IS 160-177
- LITERALS ASSEMBLED HERE (160-163).
- THEY ARE SHOWN AT THE END OF THE LISTING.

**BINDARY CARD CCG001**

- GCG000 CCG000 CCG000 GCG000 CCG000
- GCG000 GCG000 100000 100001
- GCG000 GCG000 G10000 100001 CINIT PIE
- DISFIL DISPLAY FILE START
### BINARY CARE CCCC3

<table>
<thead>
<tr>
<th>INSTRUCTIONS</th>
<th>PCINTR RELORG</th>
<th>TEMPORARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>G401</td>
<td>0 CCCC 00000 10000 COMSTO PZE</td>
<td>COMPUTE AND STORE COMPONENT</td>
</tr>
<tr>
<td>G402</td>
<td>10CC 010 0041010001 TAD =0-762 10CC-2*7</td>
<td></td>
</tr>
<tr>
<td>G403</td>
<td>00CC 00000 00750010000 SMA</td>
<td></td>
</tr>
<tr>
<td>G404</td>
<td>10CC 050 0040100001 JMP **3 POSITIVE</td>
<td></td>
</tr>
<tr>
<td>G405</td>
<td>00CC 00000 00740100001 CIA MAKE SIGN-MAGNITUDE</td>
<td></td>
</tr>
<tr>
<td>G406</td>
<td>1CC 00000 00410100001 TAD =0CC0</td>
<td></td>
</tr>
<tr>
<td>G407</td>
<td>10CC 030 0060100001 CCA TEMP2 SAVE COMPONENT</td>
<td></td>
</tr>
<tr>
<td>G408</td>
<td>10CC 010 0040100001 TAD COMSTG GET ARGUMENT</td>
<td></td>
</tr>
<tr>
<td>G409</td>
<td>10CC 010 00040010001 CCA POINTR</td>
<td></td>
</tr>
</tbody>
</table>

### BINARY CARE CCCC4

<table>
<thead>
<tr>
<th>INDIRECT ADDRESSING</th>
<th>PCINTR RELORG</th>
<th>TEMPORARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC412</td>
<td>06CC 00000 0713010000 STL RAR INTENSIFY OR ESCAPE BIT</td>
<td></td>
</tr>
<tr>
<td>CC413</td>
<td>06CC 00000 0060110001 TAD TEMP2</td>
<td></td>
</tr>
<tr>
<td>CC414</td>
<td>06CC 00000 06211100001 CDF B DATA FIELD = 1</td>
<td></td>
</tr>
<tr>
<td>CC415</td>
<td>10CC 0630040010001 CCA* POINTR STORE COMPONENT</td>
<td></td>
</tr>
<tr>
<td>CC416</td>
<td>06CC 00000 06201100001 CUF 0 DATA FIELD = 0</td>
<td></td>
</tr>
<tr>
<td>CC417</td>
<td>10CC 020 0040100001 ISZ COMSTG ADJUST EXIT POINT</td>
<td></td>
</tr>
<tr>
<td>CC418</td>
<td>10CC 06500040110001 JMP* COMSTG EXIT</td>
<td></td>
</tr>
</tbody>
</table>

### BINARY CARE CCCC5

<table>
<thead>
<tr>
<th>PCINTR RELORG</th>
<th>TEMPORARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>3414</td>
<td>10000 DISFIL SCI INT+4 SET SCALE AND INTENSITY</td>
</tr>
<tr>
<td>3415</td>
<td>10000 EUS POINT+CSB INTENSIFY POINT AT CENTER</td>
</tr>
<tr>
<td>3416</td>
<td>0000005CC 10000 CCT 3000,5000</td>
</tr>
<tr>
<td>3417</td>
<td>00000050CC 10000 EUS VEC BUBBER-BAND VECTOR</td>
</tr>
<tr>
<td>3418</td>
<td>000000112110000 VEY CCT 3000</td>
</tr>
<tr>
<td>3419</td>
<td>000000114110000 VEYX CCT 4000</td>
</tr>
<tr>
<td>3420</td>
<td>000000276710000 CCT 2767 LEFT AND DOWN DARK</td>
</tr>
<tr>
<td>3421</td>
<td>0000000051610000 SG2 INT+6 SET SCALE AND INTENSITY</td>
</tr>
<tr>
<td>3430</td>
<td>000000114110000 SVEG EDS</td>
</tr>
<tr>
<td>3431</td>
<td>000000276710000 CCT 2767 LEFT AND DOWN DARK</td>
</tr>
</tbody>
</table>
MACRO CALL

SIDE 4057,2140 RIGHT
EDS SVEC
OCT 4057
SLPSI CLPSI
EDS SVEC
OCT 2140
SIDE 5740,0041 UP
EDS SVEC
OCT 5740
SLPSI CLPSI
EDS SVEC
OCT 0041
SIDE 4077,0140 LEFT
EDS SVEC
OCT 4077
SLPSI CLPSI
EDS SVEC
OCT 0140
SIDE 7740,0061 DOWN

ASSEMBLED CODE FOR JUMP COMMAND

LITERALS
NEGATIVE OCTAL LITERAL
SPECIFIED AT LOCATION 402
## REFERENCES TO DEFINED SYMBOLS

<table>
<thead>
<tr>
<th>VALUE</th>
<th>NAME</th>
<th>STATEMENT NUMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC163</td>
<td>CINIT</td>
<td>68</td>
</tr>
<tr>
<td>CC401</td>
<td>CCSTGC</td>
<td>96,99,114,121,122</td>
</tr>
<tr>
<td>LC600</td>
<td>DISFIL</td>
<td>73,144.0001,144.0005</td>
</tr>
<tr>
<td>CC600</td>
<td>.OCC1</td>
<td>51.0002</td>
</tr>
<tr>
<td>CC601</td>
<td>.OCC2</td>
<td>52.0002</td>
</tr>
<tr>
<td>CC1CC</td>
<td>.OCC3</td>
<td>64.0003</td>
</tr>
<tr>
<td>CCCCC</td>
<td>.OCC4</td>
<td>64.0005</td>
</tr>
<tr>
<td>CC1CC</td>
<td>.OCC5</td>
<td>64.0005</td>
</tr>
<tr>
<td>CC160</td>
<td>.OCC6</td>
<td>70.0003</td>
</tr>
<tr>
<td>CC104</td>
<td>.OCC7</td>
<td>70.0005</td>
</tr>
<tr>
<td>CC160</td>
<td>.OCC10</td>
<td>70.0005</td>
</tr>
<tr>
<td>CC16C</td>
<td>.OCC11</td>
<td>71.0003</td>
</tr>
<tr>
<td>CC164</td>
<td>.OCC12</td>
<td>75.0004</td>
</tr>
<tr>
<td>CC226</td>
<td>.OCC13</td>
<td>75.0004</td>
</tr>
<tr>
<td>CC226</td>
<td>.OCC14</td>
<td>76.0003</td>
</tr>
<tr>
<td>CC457</td>
<td>.OCC16</td>
<td>104.0006</td>
</tr>
<tr>
<td>CC260</td>
<td>.OCC17</td>
<td>104.0008</td>
</tr>
<tr>
<td>CC400</td>
<td>.OCC20</td>
<td>104.0008</td>
</tr>
<tr>
<td>CC400</td>
<td>.OCC21</td>
<td>105.0005</td>
</tr>
<tr>
<td>CC401</td>
<td>.OCC22</td>
<td>105.0005</td>
</tr>
<tr>
<td>7C620</td>
<td>.OCC23</td>
<td></td>
</tr>
<tr>
<td>CC421</td>
<td>.OCC24</td>
<td>127.0010</td>
</tr>
<tr>
<td>LC600</td>
<td>.OCC25</td>
<td>127.0010</td>
</tr>
<tr>
<td>1C600</td>
<td>.OCC27</td>
<td>144.0004</td>
</tr>
<tr>
<td>1C642</td>
<td>.OCC31</td>
<td>145.0003</td>
</tr>
<tr>
<td>CCC00</td>
<td>LCC.C</td>
<td>51.0002,52.0002,64.0003,70.0003,75.0002,104.0001,104.0006,105.0006</td>
</tr>
<tr>
<td>1C0400</td>
<td>.START</td>
<td></td>
</tr>
</tbody>
</table>

## REFERENCES TO LOCATION COUNTERS

<table>
<thead>
<tr>
<th>LC START</th>
<th>NAME</th>
<th>STARTING AND ENDING STATEMENT NUMBERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCCCC</td>
<td>1-64.0003,64.0003-7C.0003,7C.0006-71.0001,71.0004-75.0002,75.0004-104.0009-105.0003,105.0006-127.0008,127.0011-145.0001</td>
<td></td>
</tr>
<tr>
<td>4CC00</td>
<td>//</td>
<td>64.0004-64.0004,70.0004-7C.0005,71.0002-71.0003,75.0003-75.0004,104.0007-104.0008,105.0004-105.0005,127.0009-127.0010,145.0002-1</td>
</tr>
<tr>
<td>PAGE STARTING ADDRESS</td>
<td>LINKED INSTRUCTION ADDRESS</td>
<td>LINK WORD ADDRESS</td>
</tr>
<tr>
<td>-----------------------</td>
<td>----------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>000000</td>
<td>00102</td>
<td>00164</td>
</tr>
<tr>
<td>002000</td>
<td>00230</td>
<td>00255</td>
</tr>
<tr>
<td>004000</td>
<td>00407</td>
<td>00167</td>
</tr>
</tbody>
</table>

**FINAL STATUS OF THE INITIAL PAGE OF FIELD 0**

| 100000                | 4                          | 4                | 8               |                                                 |        |
| 106000                |                            |                  |                 |                                                 |        |
23.2 Explanation

ASSEMBLY

In the above assembly listing of RUBBAN the reader should note:

1. The 49 cards composing the DEC-338 PDPMAP DEFINITIONS were included at the beginning of the source deck, but only the first card is listed.

2. The program includes the definition and use of a new Operate instruction mnemonic. The OCTADR pseudo-operation was used twice.

3. A macroinstruction was defined and was used four times.

4. The four origin pseudo-operations were used: OCTORG, ABSORG, RELORG, and PAGE (with and without an argument). Also the FIELD pseudo-operation was used.

5. Positive and negative octal literals were used and pooled together on page zero by the LORG pseudo-operation.

6. Operate instructions and DEC-338 commands were used alone and in double and triple combinations.

7. Input/Output instructions were used alone, and, in the case of CDF, a decimal number was included to form the proper instructions.

8. The uses of "*" as comment card indicator, indirect addressing indicator, and current location indicator were all included.

9. Examples of the PZE and OCT data generation pseudo-operations are given.

10. There are some off-page memory references, uses of the LINKS pseudo-operation, and an example of a linking option.
CROSS-REFERENCE DICTIONARY

The page following the assembly listing is the Cross-Reference Dictionary of RUBBAN produced by IBMAP. It is a symbol table in lexicographic order along with the numbers of the statements where each symbol was referenced.

When a symbol table is obtained, all symbols listed in the Cross-Reference Dictionary will be punched by SYMTAB except:

1. LOC.0
2. Symbols of the form ..nnnn (created symbols)
3. Symbols of the form P.nnnn (literal symbols)

POSTPROCESSING (LINKING)

As a result of the assembly of RUBBAN, two linking areas were specified through uses of the LINKS pseudo-operation:

AREA A: locations 160 - 177. Note that locations 160 - 163 can be used for linking, and locations 164 - 177 can be used for the creation of links.

AREA B: locations 226 - 377. Note that those locations between 226 and 257 where memory reference instructions were not assembled can be used for linking, and locations 260 - 377 can be used for the creation of links.

A step-by-step account of the linking procedure is given below: The reader is advised to follow the PDPMAP Postprocessor output in parallel with the explanation.

1) The first instruction which requires a link is at location 102, where IBMAP assembled a JMP instruction addressing location 226. Since no location in the linking area A already contains 0226, a link must be created, and the first possible link word is 164.
Therefore the instruction in location 102 is made into an indirect JMP
instruction addressing location 164, which in turn contains 0226.

2) Similarly, a link is created for the JMP instruction in location 103.

3) The summary for the page indicates that two links were created for
instructions on the first page and linking area A has 10 (decimal) words
available for the creation of further links.

4) The next instruction requiring a link is at location 230, where there
is a cross-field DCA instruction addressing location 10606. Since the linking
area B includes a location containing 0606, no link creation is necessary. Instead,
location 255 is used as a link by forming at location 230 an indirect DCA instruction
addressing location 255. The PDPMAP Postprocessor listing includes the number
0606 in parentheses to indicate an already existing link word was used.

5) Similarly, a link was used at location 252 for the DCA instruction in
location 231.

6) Location 247 contains the next memory reference instruction which requires
linking. Since there is no existing link word on the current page, an attempt
is made to use a link on the initial page. The Postprocessor finds that there
is such a link work in location 163, and therefore it is not necessary to
create one.

7) The JMS instruction at location 251 causes a link to be created on the
current page. The link is placed at the first available location in linking
area B.

8) As at location 247, an existing initial page link is used at location 253.

9) The instruction assembled at location 254 is an example of a user-controlled
linking option. If the option had not been included, the Postprocessor would have
used the link previously created on the current page. The option, however, forces
initial page linking and a link is created on the initial page.
10) The summary of results of the current page are listed. Note that the number of links created represents only those links created on the current page (in linking area B).

11) There is no linking area on the same page as location 407, and therefore a link is created on the initial page.

12) The link created above is used by the instruction at location 413.

13) There are no summary statistics listed for this page since no linking area was specified.

14) A summary for the field is then listed indicating the final number of unused locations in the linking area on the initial page. This number is two less than the number originally listed for that linking area since two links were created as the Postprocessor proceeded through the field.

15) Since there are no further occurrences of the LINKS pseudo-operation, no more summaries are included. A page starting address is printed for each page where code was assembled. In addition, the initial address of a memory field is always listed if there is assembled code somewhere on that field.
REFERENCES


This report describes the PDPMAP Assembly System which is used to assemble symbolic programs written for a Digital Equipment Corporation PDP-8 or DEC-338 with up to 16-K memory locations. The system is implemented at the University of Pennsylvania on an IBM 7040 and DEC PDP-8 connected by a high-speed data channel (IBM 7904 and DEC DMO3). The PDPMAP System uses the powerful assembler of a larger computer (IBM 7040 MAP Assembler) to quickly assemble programs for a small computer.
Assembly System
Microprogramming
Microinstructions
Off-page linking