UNIVAC
DATA PROCESSING DIVISION
1050
CARD SYSTEM

ASSEMBLY SYSTEM

CARD ASSEMBLY SYSTEM

REFERENCE MANUAL

UP-3915-1.01

This manual is published by the UNIVAC Division of Sperry Rand Corporation in loose leaf format as a rapid and complete means of keeping recipients apprised of UNIVAC ® Systems developments. The UNIVAC Division will issue updating packages, utilizing primarily a page-for-page or unit replacement technique. Such issuance will provide notification of hardware and/or software changes and refinements. The UNIVAC Division reserves the right to make such additions, corrections, and/or deletions as in the judgment of the UNIVAC Division, are required by the development of its respective Systems.

PRINTED IN U.S.A.

CONTENTS

ILLUSTRATIONS

FIGURES

1. INTRODUCTION

The PAL Card Assembler is a program that accepts PAL instructions and PAL assembler directives as input, and generates machine language coding as output. The PAL instruction repertoire and data generation directive have already been described in the Central Processor Manual. The card system assembler directives described in this manual are as follows:

BEGIN
ORIG
EQU
AREA
–
END

These directives are used by the programmer to communicate with the assembler program. Through them the programmer can control the disposition of the program instructions and storage areas.

Once the program is written and punched into cards it is ready to be assembled by the PAL Card Assembler along with the selected UNIVAC software input/output control routines. Section 3 describes, step by step, the operating procedure for assembling the source code programs. Error condition procedures are also explained.

Finally, Section 4 describes the operation of the Load Routine and the operating procedure for loading the assembled program.

2. ASSEMBLER DIRECTIVES

Assembler directives are communication devices used by the programmer to supply information to the PAL Assembler. The information supplied by these directives controls the disposition of the program, program instructions, or storage areas required by the program. Although they affect the assembly of program instructions, the assembler directives are not assembled as program instructions.

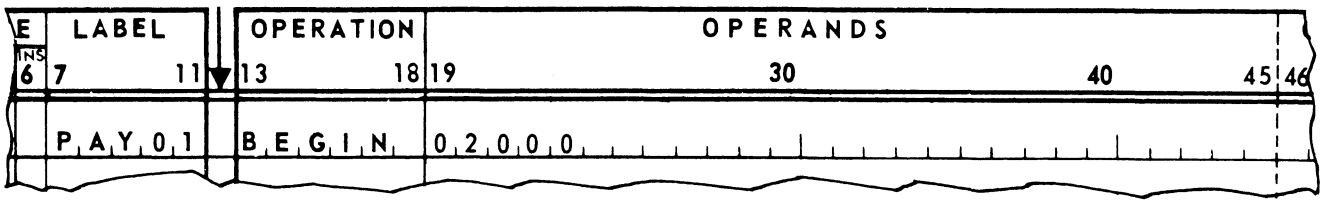
Assembler directives effect operations such as starting and ending the assembly process; relating different segments of a program that do not occupy contiguous locations; and locating the input/output and working storage areas relative to program instructions.

2.1. BEGIN and END

The directives BEGIN and END cause the start and termination, respectively, of the assembly process. Every program to be assembled must have the assembler directive BEGIN in the operation field of its first line and the assembler directive END in the operation field of its last line.

Columns 7 through 12 of the BEGIN line may contain the program name. If this is done, the program name will be punched in columns 75 through 80 of the output object code deck.

The operand field of the BEGIN line must contain a single octal or decimal integer. For example, if the programmer writes the following:



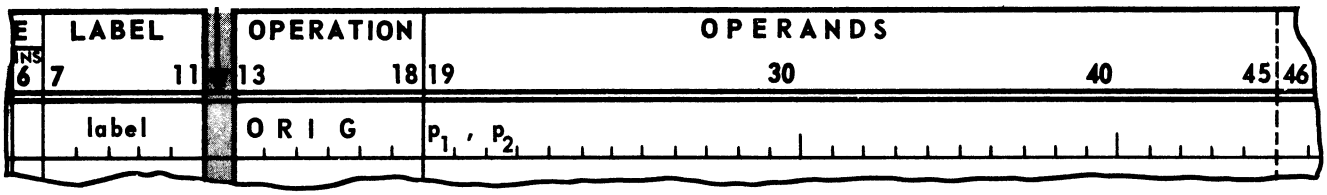
the address of the first character of the program called PAY01 is octal 2000.

The assembler directive END must appear in the operation field of the last line of the program. This directive terminates the assembly.

The operands field of the END line must contain the label of the first object program instruction to be executed when the program is loaded. The assembled object program deck will have, on the last card, a jump to the address generated from the label.

2.2. ORIG

The ORIG directive causes the assembler location counter ($) to be reset to a value determined by 1 and 2. It enables the programmer to assign the location to be used by the instruction or assembler directive on the following line:



The label, if present, is the symbolic name of the designated location.

The operands field may contain one or two parameters:

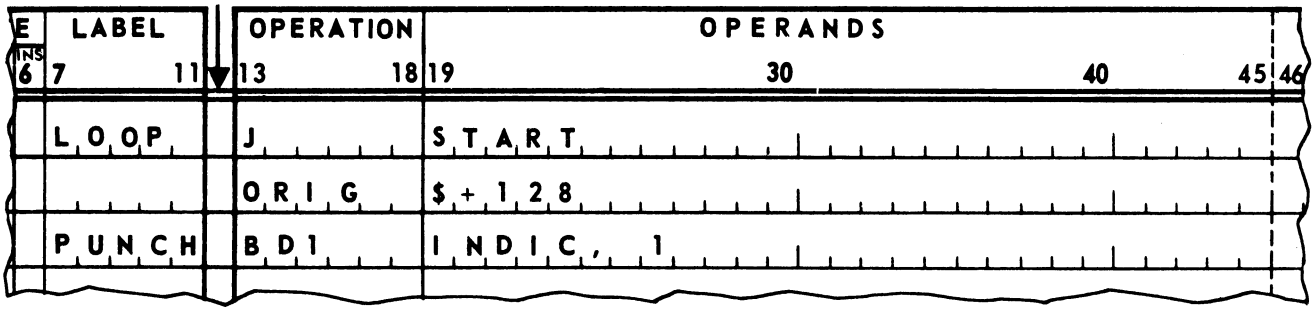
p_1 may be

- a previously defined label, with or without an address modifier,
- $, the current value of the location counter, with or without an address modifier (this modifier may be + or −), or
- an octal or decimal address.

This parameter assigns the location to be used by the instruction or assembler directive on the following line. However it can be modified by parameter p_2.

p_2 may be any number from 2–4096 that is a power of 2, i.e., (2, 4, 8, 16, ...4096). If present, the address of the next location is the first, higher than that designated by parameter p_1, that is an integral multiple of P_2.

To illustrate the ORIG directive, assume that the programmer wishes to leave 128 locations free between two coded lines. The ORIG line is written as follows:



The label PUNCH is 128 positions beyond the address of the last character of the line LOOP.

The same result could be obtained as follows:

```
E  LABEL      OPERATION    OPERANDS
INS
6  7      11  13     18    19        30        40        45 46
   LOOP       J            START
              ORIG         LOOP+133
   PUNCH      BDI          INDIC,1
```

Since the label LOOP names the first character of the line on which it appears, and since the line labeled LOOP is a five character instruction, the expression LOOP + 5 written on the next line would be equivalent to $. To allow 128 positions between the two lines LOOP and PUNCH, therefore, the programmer must write either $ + 128 or LOOP + 133.

If the programmer writes the following:

```
E  LABEL      OPERATION    OPERANDS
INS
6  7      11  13     18    19        30        40        45 46
   LOOP       J            START
              ORIG         $+128,64
   PUNCH      BDI          INDIC,1
```

the label PUNCH will be at least 128 positions beyond the last character of the line LOOP and will have an address that is an integral multiple of 64.

2.3. EQU

The EQU directive is used to assign the value of the expression (p_1) in the operands field to the symbol in the label field.

```
E  LABEL      OPERATION    OPERANDS
INS
6  7      11  13     18    19        30        40        45 46
   label      EQU          p1,p2
```

The operands field may contain one to three parameters:

p_1 may be

- a previously defined label, with or without a modifier,
- $, the current value of the location counter, with or without a modifier (+n or −n),
- a decimal or octal absolute address, or
- a decimal or octal constant (with a maximum value of 65, 535).

p_2 if present, is a decimal or octal number specifying the number of characters in the field being defined. The value of this expression may not exceed 16.

p_3 is a number from 1 to 7 specifying an index register that the assembler is to associate with the defined label. The value of this expression is inserted into the index register portion of any instruction referencing the defined label, unless an index register expression is already supplied in that instruction.

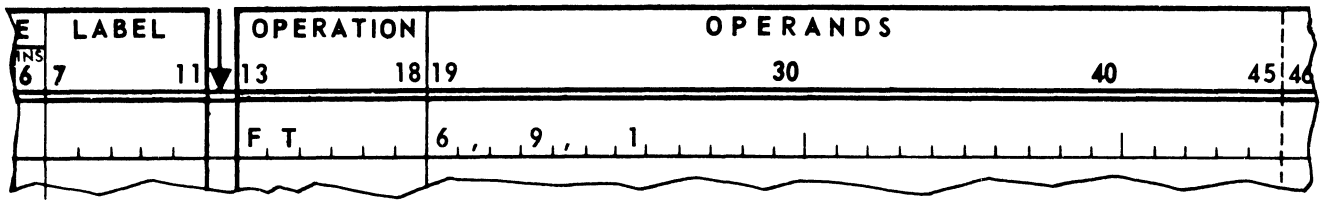
The EQU directive is useful both as an aid to good documentation and as a programming convenience. For example, the index registers may be assigned symbolic names as follows:

E INS 6	LABEL 7–11	OPERATION 13–18	OPERANDS 19–46
	X 1	E Q U	3 9
	X 2	E Q U	X 1 + 4
	X 3	E Q U	X 2 + 4
	X 4	E Q U	X 3 + 4
	X 5	E Q U	X 4 + 4
	X 6	E Q U	X 5 + 4
	X 7	E Q U	X 6 + 4

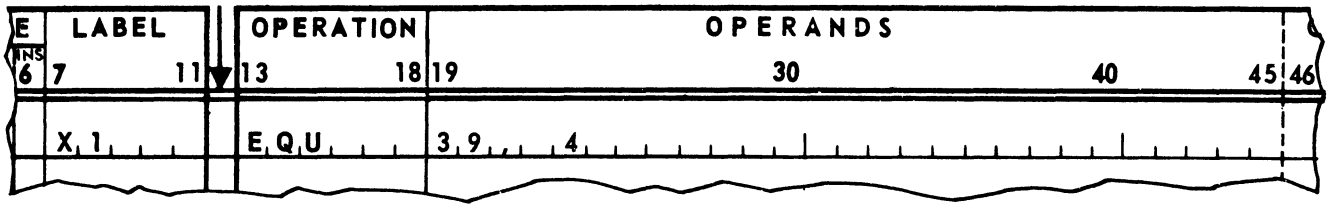
Index register one is tetrad nine, the least significant character of which is absolute location 39. Regardless of whether the symbolic names defined above will be used as M, T, or X expressions, the EQU directive or directives must equate the symbolic names to the absolute addresses of the fields. The assembler makes the necessary adjustments to convert these symbols into their appropriate values. For example, writing the following:

E INS 6	LABEL 7–11	OPERATION 13–18	OPERANDS 19–46
		F T	6 , X 1 , X 1

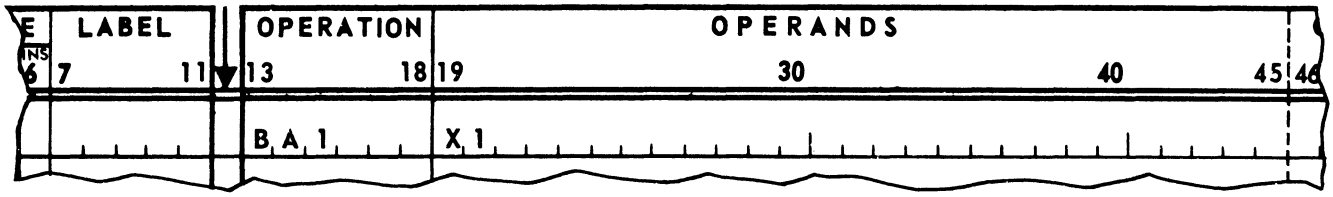
is equivalent to writing the following:



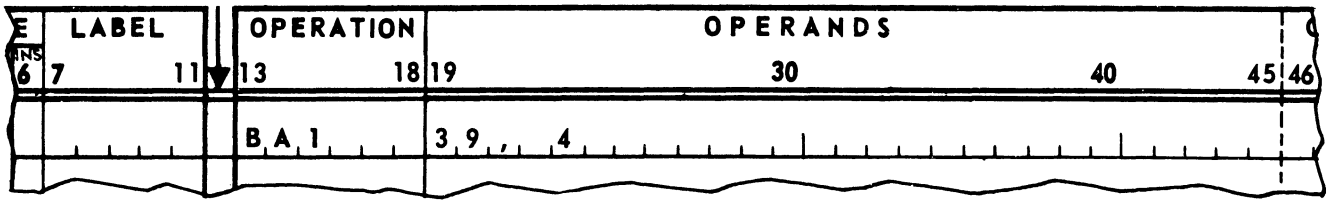
The second expression on an EQU line allows the programmer to supply the field length. The assembler will automatically insert the specified field length in every subsequent instruction addressing this field. For example, if the index register definition for index register 1 were written as follows:



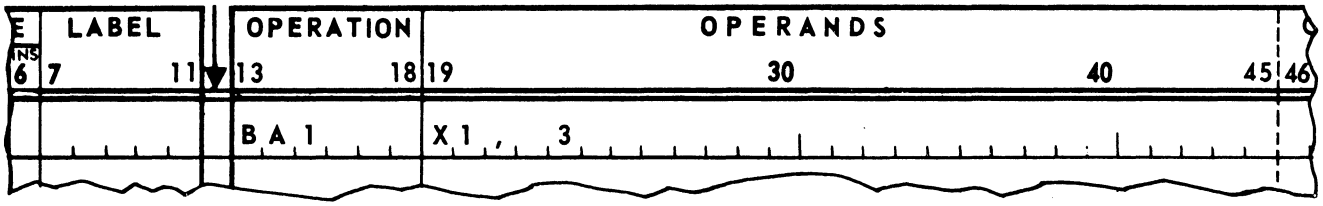
the line



is assembled as though it had been written.



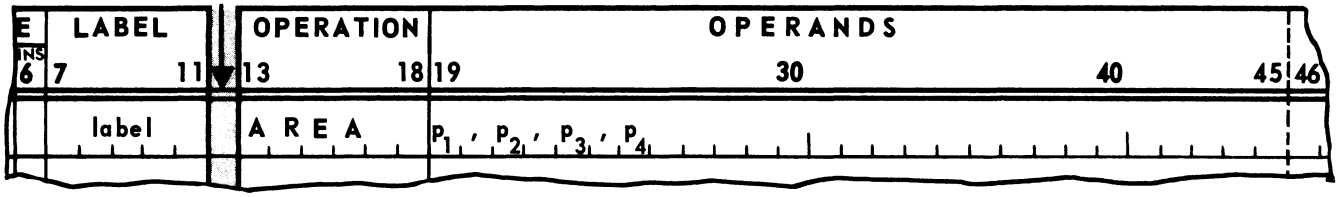
The programmer has the option to override the directive supplied field length by specifying another field length. If he writes the following:



the assembler will place only a 3 in the field length portion of this instruction. Subsequent instructions that reference X1 without specifying field length will still be assembled with a 4 in the field length portion.

2.4. AREA

The AREA directive is used to define the working storage and input/output areas.



The label of the AREA directive names the *leftmost* character of the area.

The operands field contains four parameters:

p_1 is a decimal or octal number specifying the size of the area in terms of number of characters. It must always be present.

p_2 is

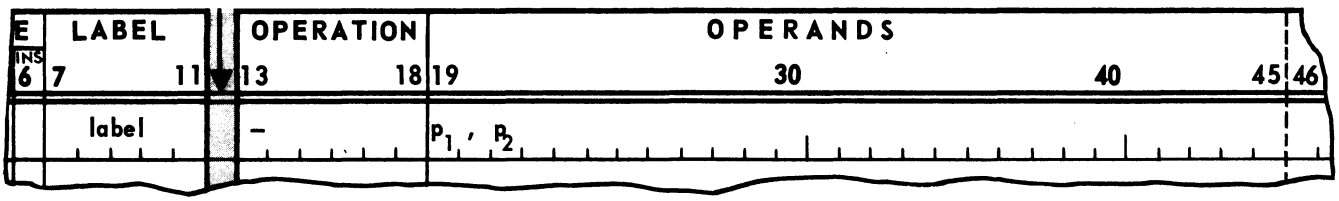
A for an area containing alphabetic data
B for an area containing binary data
I for an area containing instructions

If parameter 2 is not supplied, the area must be considered alphanumeric.

p_3 is a decimal number ranging from 0 through 63, an octal number ranging from 0 through 077, or any single character written within apostrophes, to which every character of the area is to be preset when the object program is loaded. If parameter 3 is not supplied, the area is not to be preset.

p_4 is a number from 1 to 7 specifying an index register that the assembler is to insert into the index register portion of any instruction referencing the area, or any field within the area. The programmer may elect to override this by specifying another index register. If p_4 is not supplied, any instruction referencing the area is assembled without index register modification.

Fields within the area are defined in the lines immediately following the AREA line by writing a minus sign (–) in column 13. One or two parameters are entered in the operands field.



The label of a line defining a field is the address of the rightmost character of the field.

p_1 is an octal or decimal number specifying the length of the field. This expression must always be present.

p_2 is a decimal or octal number specifying the position of the field within the area. The number specifies the position of the *rightmost* character of the field relative to the first or leftmost character of the area. This expression may be omitted if the first character of the field immediately follows the last character of the last field defined.

An example of an AREA definition follows:

E INS 6	LABEL 7 11	OPERATION 13 18	OPERANDS 19 30 40 45 46
	CARD	AREA	80,A,,,3
	CSECT	-	2
	CACNO	-	4
	CNAME	-	13,23
	CAMNT	-	5,35
	CACTN	-	1

The first line is the AREA definition, defining 80 consecutive positions, the leftmost of which is labeled CARD. The area is defined as being alphanumeric and is not to be preset to any value. All instructions referencing this area that have no index register expressions are to have 3 inserted in the index register portion.

CSECT is the name of a two character field whose rightmost character is the second character of the area.

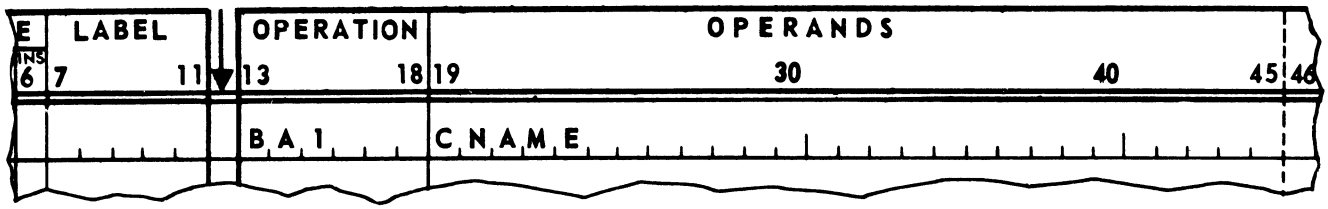
CACNO is the label assigned to the rightmost character of a four character field. CACNO names the fourth character beyond CSECT.

CNAME is the label of a 13 character field whose rightmost character is the twenty-third character of the area.

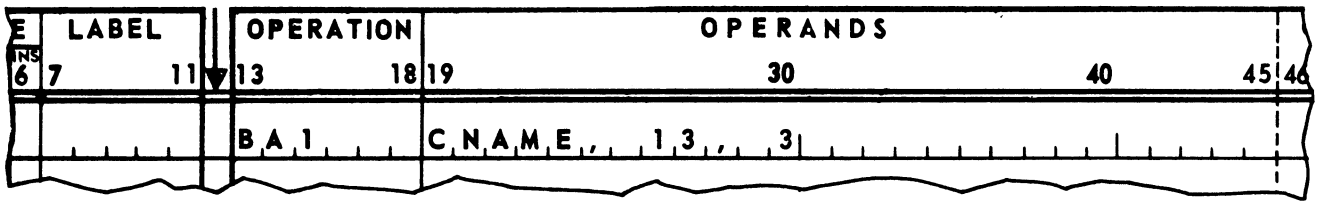
CAMNT is the label of the thirty-fifth character of the area, which is the rightmost character of a five character field.

CACTN is the label of the thirty-sixth character of the area. The field CACTN is one character in length.

As a result of this definition, the programmer may write the following:



which is assembled as though he had written the following:



The definition directs the assembler to supply field length and index register expressions automatically, unless the programmer chooses to override this automatic action by supplying different field length and index register expressions.

3. INSTRUCTIONS FOR ASSEMBLING

Once the program has been written in the Program Assembly Language (PAL) and punched into cards, it is ready for assembly. These cards, with the required I/O routines, compose the assembly deck. The output of the assembler is the program and required I/O routine punched in object code and a printer listing.

3.1. SETUP PROCEDURE

Sense switch 1 controls punching of the object deck.

OFF – Object deck will be punched.
ON – Object deck will not be punched.

Sense switch 2 controls printing of the assembly listing.

OFF – Listing will be printed.
ON – Listing will not be printed.

Sense switch 3 controls punching of the label table.

OFF – Label table will not be punched.
ON – Label table will be punched.

3.2. FIRST PASS INSTRUCTIONS

3.2.1. Card Punch

(1.) Put blank cards in the input hopper.

(2.) Depress the POWER ON and OFFLINE buttons until illuminated.

(3.) Depress the READY button, and then the MANUAL FEED button repeatedly until card(s) appear in the output stacker. Remove these card(s) from the output stacker.

(4.) Depress the OFFLINE button until the light is extinguished.

3.2.2. Printer

(1.) Depress the OFFLINE button until the button is illuminated.

(2.) Set the paper five holes above the paper feed sprocket.

(3.) Depress the OFFLINE button until the light is extinguished.

3.2.3. Card Reader

(1.) Place the first pass PAL assembler deck in the card input hopper, and follow, in sequence, with the worker program, the I/O routine area statements, the I/O routine specializing EQU cards, the I/O routine, the END card, five blank cards, and the card weight.

(2.) Depress the READY button.

3.2.4. Central Processor Console

(1.) Depress the CLEAR button.

(2.) Depress the LOAD CARD Mode button.

(3.) Depress the PROGRAM START button.

(4.) Depress the CONT Mode button.

(5.) Depress the PROGRAM START button.

(6.) Display stop 30 017777 60 indicates the end of the first pass.

3.3. SECOND PASS INSTRUCTIONS

3.3.1. Card Reader

(1.) Remove cards from the output stacker and separate the source code from the first pass PAL assembler deck.

(2.) Place the second pass PAL assembler deck, followed by the source code deck, followed by five blank cards and a weight in the input magazine.

(3.) Depress the READY button.

3.3.2. Central Processor Console

(1.) Depress the PROGRAM START button to assemble the second pass.

(2.) Display stop 30 017777 60 indicates the end of assembly.

3.4. ASSEMBLY OUTPUT

The output of the assembly process consists of an object code card deck, which is ready to be loaded and run, and a printed listing. The listing comprises a label table and a listing of the assembly program.

The label table contains seven headings under which are listed all labels and relevant data. The heading of each column and data contained is as follows:

COLUMN HEADING	CONTENT
LABEL	All the labels used in the assembled program.
ADDRESS	The address of the character labeled.
LENGTH	The length of the field labeled.
INDEX	Any index register associated with the label.
TYPE	The type data in the field (A for alphanumeric, B for binary, I for instruction).
LOAD KEY	0 for an absolute label address, and 1 for a relative label address.
ERROR	Error code, if any. (See the L error code in the list below.)

The assembled program listing follows the label table. It contains eleven headings under which are listed the lines of coding, represented in octal on the left hand side, and PAL on the right. The heading of each column and the data contained are as follows:

COLUMN HEADING	CONTENT
ERROR	Error code, if any. (See list of error codes.)
LOCATION	The octal address of the instruction or field listed. An asterisk to the left of the location indicates a jump instruction.
SIZE	Length of area or field in characters.
IX	Index register associated with the area or field.
DATA	For a data line, up to 16 alphanumeric characters, representing the data itself. For an instruction line, the instruction in octal code (the format is given in Section 1.4 of the Central Processor manual). For an XF instruction line, the instruction in octal code (the format is given on the UNIVAC 1050 Code Card, UP 3930 Rev. 1).
LINE	The line of coding in alphanumeric characters as written in PAL.
LABEL	
OP	
OPERANDS	
COMMENTS	
PROGRAM ID	The name of the program in alphanumeric characters.

LABEL	ADDRESS	LENGTH	INDEX	TYPE	LOAD KEY	ERROR
REA	000600				0	
PGM	001040				0	
XHC	077777				0	
CHR	000077				0	
C1	000577				0	
C2	001061				0	

Figure 1. Sample Label Table Listing

```
ERROR  LOCATION  SIZE IX   DATA          LINE  LABEL OP    OPERANDS                    COMMENTS                                  PROGRAM-ID
         000600                         00101 LD8X  BEGIN 0600
         000600                         00102 REA   EQU   0600                                                                  LD8X00
         001040                         00103 PGM   EQU   REA+160                                                              LD8X00
         077777                         00104 XHC   EQU   077777                                                               LD8X00
         000077                         00105 CHR   EQU   077                                                                  LD8X00
         000577                         00106 C1    EQU   REA-1                                                                LD8X00
         001061                         00107 C2    EQU   REA+177                                                              LD8X00
                                        00108 .           FILLS MEMORY TO THE CHARACTER SPECIFIED BY CHR, INCL.              LD8X00
                                        00109 .                TETRAD AREA (EXCEPT T3,T8,T16,T18,T19) AND                    LD8X00
                                        00110 .                0520 TO  XCH  EXCEPT FOR REA TO REA+157.                      LD8X00
                                        00111 .           USES  160 LOCATIONS AT REA, 85  AT PGM.                            LD8X00
         000440                         00115       ORIG  0440                                                                 LD8X00
    *    000440  3000041060  00120       JC    0410,060                                                                     LD8X00
         000447       000410 00125       +3    0410                                                                         LD8X00
         000450  5600044730  00130       BA2   0447, 8                                                                      LD8X01
         000455  6000050730  00135       AC    0507, 030                                                                    LD8X01
         000462  5200041730  00140       SA2   0417, 8                                                                      LD8X01
    *    000467  3000052036  00145       JC    0520,30                                                                      LD8X00
         000475        0077  00150       +2    CHR                                                                          LD8X00
         000477        0077  00151       +2    CHR                                                                          LD8X00
    *    000500  3000050060  00155       JC    0500,060                                                                     LD8X00
         000507       000450 00160       +3    0450                                                                         LD8X00
    *    000510  3000051060  00165       JC    0510,060                                                                     LD8X00
         000517       000510 00170       +3    0510                                                                         LD8X00
         000520  5600047701  00175       BA1   0477,1                                                                       LD8X00
         000525  5200001617  00178       SA1   14,15                                                                        LD8X00
         000532  2600022710  00181       PD    0227, 8                                                                      LD8X01
         000537  2600011700  00184       PD    0117, 16                                                                     LD8X01
         000544  5200043700  00187       SA1   0437,16                                                                      LD8X00
         000551  2000060044  00190       FT    REA,36                                                                       LD8X00
         000556  2000060045  00193       FT    REA,37                                                                       LD8X00
         000563  4020610000  00196       XF    061,0,0,1                                                                    LD8X00
    *    000570  3000064000  00199       JC    REA+32,0                                                                     LD8X00
                                        00205 .           DELETE CARDS FOR PAGE 2 IF TETRAD FILL ONLY.                       LD8X00
                                        00206 .           DELETE CARDS FOR PAGE 3 IF MEMORY FILL ONLY.                       LD8X00
                                        00207 .           DELETE CARDS FOR PP. 2,3 IF NEITHER MEM FILL OR TETRAD             LD8X00
                                        00208 .                FILL.                                                         LD8X00
```

Figure 2. Sample Assembled Program Listing

The error codes for both the label table and program listing are as follows:

ERROR CODE	DESCRIPTION
L	Duplicate or undefined label.
E	Expression is too large or has been omitted.
O	Unrecognizable OP code.
S	Card sequence error.
C	Number of cards processed in the first pass does not agree with the number of cards processed in the second pass (can appear only on the END line).

3.5. ERROR PROCEDURES

DISPLAY	PASS	DESCRIPTION	ACTION
30 017771 60	1 & 2	No BEGIN card	Refeed source deck with a valid BEGIN card. Depress the PROGRAM START button to continue assembly.
30 010077 60	1	Label table exceeded	Depress the PROGRAM START button to continue assembly. All labels that exceed the table will appear on the output listing with "L" errors. The number of labels by which the label table was exceeded will be printed at the end of the label table printer listing. After assembly use the $ option to reduce the number of labels.
30 110000 60	2	Reader Error	Refeed cards in error stacker. Depress the CLEAR button on the reader. Depress the PROGRAM START button.
30 11000x 60	1	Reader Error	Replace x cards in the input magazine. If x does not match the number of cards in the error stacker, remove the difference from the normal stacker. Depress the CLEAR button on the reader. Depress the PROGRAM START button.
30 120000 60	2	Punch error	Depress the CLEAR button on punch unit. Depress the PROGRAM START button. All cards selected into the error stacker may be discarded.
30 100000 60	2	Printer error	Return printer to normal condition. Depress the CLEAR button on the printer. Depress the PROGRAM START button.
30 017777 60	1 & 2	Completion	Pass in operation is completed.
30 070105 60	All	Card count error during load of assembler	The assembler object card deck does not contain the proper number of cards. Restart pass in which error occurred using a complete assembler deck.

4. INSTRUCTIONS FOR LOADING

4.1. THE LOAD ROUTINE

The Load Routine performs the following functions in the sequence listed (optional modes of operation are described in Sections 4.2 and 4.3):

- Establishes the Class I, Class II, and card reader Class III interrupt entries.
- Fills storage with a specified character (normally ¤).
- Loads itself into consecutive locations immediately following the Load Routine input area. The label of the first character of the Load Routine is PGM; the label of the reader input area is REA.
- Loads the assembled worker program through the loader input area.

The only locations that cannot be loaded by the load routine are tetrads 7, 8, 16, 18, 19, 36, and 37, the card reader interrupt entry, the loader input area (REA), and the load routine area (PGM).

The 80 column Card Reader Load Routine occupies 85 characters, and its input area occupies 160 characters.

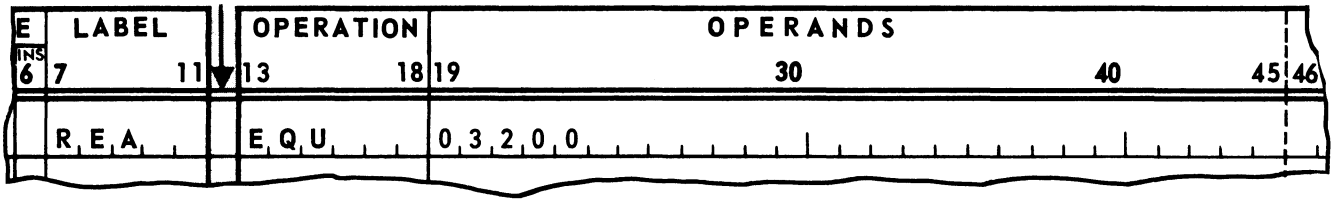
The 90 column UNIVAC 0704 and 0706 Card Reader Load Routine occupies 105 characters, and its input area occupies 109 characters.

The 90 column UNIVAC 0701 Card Reader Load Routine and input area occupy 90 characters each.

4.2. ASSEMBLING THE LOAD ROUTINE

At the beginning of the source code Load Routine deck are six EQU directive cards. The first four cards (with labels REA, PGM, XHC, and CHR) may be altered to change the routine as follows:

- To change the location of the loader input area, redefine REA as equivalent to the new address. The address supplied must be a multiple of 128 (or 0200 if expressed in octal notation). For example, if the loader input area were to begin at location 03200, the card defining REA would be replaced with the following:



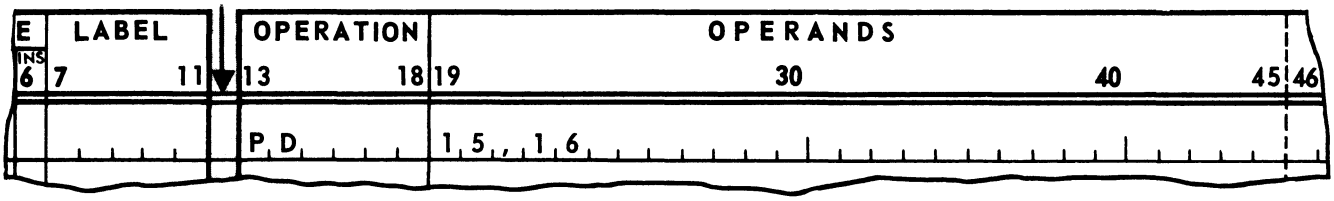
  The length of the area depends upon the type card reader used (see Section 4.1).

- To change the location in which the Load Routine is to be stored, redefine PGM as equivalent to the address of the first location the load routine is to occupy. The size area occupied by the Load Routine also depends upon the type card reader used (see Section 4.1).

- To set an upper limit to the amount of storage that is to be cleared and set to the specified character (CHR), redefine XHC as equivalent to the last character to be changed.

- To change the character with which storage is to be filled, replace the definition of CHR with the character desired. The character is written as a constant in the PAL convention.

The tetrad area of storage can be cleared to blanks by replacing the card for line 00310 with the following line:



Further Load Routine options are available when loading the assembled worker program (see Section 4.3).

Use the Card Assembler and Card Assembler operating procedure described in Section 3 to assemble the Load Routine. The output of assembly will be eight object cards. Discard the first and last cards (R and T cards).

4.3. LOADING OPTIONS

Once the Load Routine has been assembled three further options are available.

The Load Routine may be used without clearing and setting the tetrand area and storage to the designated character. To do so, remove the second and third cards from the assembled loader deck.

The Load Routine may be used to set the tetrad area to the designated character, but not storage. To do so, remove the second card from the assembled loader deck.

The Load Routine may also be used without the check sum feature. To do so, remove the fifth card from the assembled loader deck. This reduces the space required for the load routine by 35 characters.

4.4. OPERATING PROCEDURE

To load the assembled worker program, place the object Load Routine, followed by the worker program, in the card reader. The object loader will consist of three to six cards, depending upon the options (Section 4.3).

At the Central Processor Console

(1.) depress the CLEAR button,

(2.) depress the LOAD CARD Mode button,

(3.) depress the PROGRAM START button,

(4.) depress the CONT Mode button, and

(5.) depress the PROGRAM START button.

The load routine will load itself, fill storage with the specified character, load the worker program, and transfer control to the program loaded.

4.5. ERROR PROCEDURES

DISPLAY	ERROR	ACTION
30 0xxxxx 60	Reader Error	If an error occurs while reading the Load Routine, the computer will stop with xxxxx in the M portion of the display indicating an address in the load input area. Restart the load procedure from the beginning.
30 110000 60	Reader Error	If the channel one abnormal light is lit, refeed the card that is in the error stacker. If the channel one abnormal light is not lit, refeed the last card in the normal output stacker. Depress the CLEAR button on the card reader. Depress the PROGRAM START button.
30 11000x 60	Reader Error	Replace x cards in the input stacker. If x does not match the number of cards in the error stacker, remove the difference from the normal stacker. Depress the CLEAR button on the reader. Depress the PROGRAM START button.

UNIVAC

DIVISION OF SPERRY RAND CORPORATION

UP-3915-1.01