ALTAIR 680B CROSS-ASSEMBLER

ALTAIR 680B CROSS-ASSEMBLER

by

PRADHEEP S. KUMAR B. Tech.

A Project

Submitted to the School of Graduate Studies

in Partial Fulmilment of the Requirements

for the Degree

Master of Science

McMaster University

February 1978

MASTER OF SCIENCE (1978)
(Computation)

McMaster University
Hamilton, Ontario

TITLE: Altair 680b Cross-Assembler

AUTHOR: PRADHEEP S. KUMAR B.Tech. (Indian Institute of Technology, Madras, India)

SUPERVISOR: Professor T.J. Kennet

NUMBER OF PAGES: vi, 289

ABSTRACT

The Altair 680b Cross-Assembler is a program written in the Nova Assembly Language. It can be used to assemble Altair 680b assembly language programs. The object code can be punched on paper-tape for execution on the Altair 680b microcomputer.

This report describes the design and working of the Cross-Assembler. A program listing and a few sample runs are also included.

ACKNOWLEDGEMENTS

I wish to express my appreciation to my supervisor Dr. T.J. Kennet for his guidance and assistance during the course of this project. My special thanks to K. Chin and T. Snider for the helpful discussions during the implementation of this project.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Machine Language

"Machine Language" is the basic instruction set of the computer. Each instruction is represented by a pattern of 0's and 1's, which direct the computer, through logic circuits, to perform some action.

Writing a program using instructions of this form can be an extremely tedious process. The programmer has to specify the correct machine code for the instruction, assign locations (or addresses) to every instruction and data-element in the computer memory, and then use these addresses in the instructions. Apart from the difficulty of keeping track of a lot of numbers, it is almost impossible to either understand or make changes to the program, without expending considerable effort.

1.2 Assembly Language

Attempts were made to design better ways of writing computer programs. This led to the development of symbolic languages or assembly languages.

Assembly languages relieve the programmer of tedious book-keeping, and at the same time, allow him to make use of the computer efficiently. In [1] using an assembly language, the programmer refers to the locations of instructions, data and constants, as well as the instructions, by symbolic names. Symbols may be assigned in an arbitrary manner; when

reference is to be made to locations having these names, the symbols are used. Mnemonic symbolism is convenient; i.e. the symbols are suggestive of the function of the instruction, such as LDA (for Load Accumulator A).

An assembly language program is made up of instructions.[2] An instruction can be regarded as being made up of a number of fields; the simplest form would be two fields, opcode and address. To this a label can be added, and possibly a descriptive comment. In a fixed format scheme, the various fields are recognized by their position on the line, while in a free format scheme, they are separated by delimiters. There are many variants of the scheme. A common one is a semi-fixed format in which the label occupies a fixed field, while the rest of the fields are in free format.

The[1] task of converting symbolic operation-codes and addresses is completely straight-forward. For operations, usually the only required process is that of looking up a table for each symbol and replacing it with the numerical operation-code. For the addresses, the process is somewhat more complicated since the choice of symbols is up to the programmer, but it is still essentially a replacement problem. The computer itself could be programmed to do the translation. A program that does this job is called an ASSEMBLER.

1.3 Overview of the Cross-Assembler

This report describes an assembler for the ALTAIR 680b micro-computer, which is based on the Motorola M6800 micro-processor chip. It

is called a "CROSS-ASSEMBLER" since it is implemented on a different computer; i.e. the Data-General Nova.

The motivation behind the project was to develop an assembler to enable effective use of the ALTAIR micro-computer. Since the micro-computer was equipped with only 1k x 8 RAM, it was decided that a Cross-Assembler should be developed on the Nova minicomputer. Again, since the Nova minicomputer available was not equipped with any high level language , the Cross-Assembler was written in the Nova Assembly Language.

The Cross-Assembler is a two pass assembler. In the first pass, the source program is read, one line at a time, and a value of the "location-counter" is associated with each instruction. The length of an instruction is determined from the opcode and the operands, and this is used to update the location counter. If a label is present, it is entered into a symbol table, along with the location-counter value. Some pseudo-instructions (such as ORG, RMB) are processed completely in the first pass, while the rest are partially processed so as to allocate storage space.

The actual machine code and source-listing are produced in the second pass. Again each source line is read, and using the information gathered in pass-1, the object code is assembled. This is written out onto the object-code buffer, which is periodically flushed. The source line along with other useful information (location, value, etc.) is printed out. At the end of the second pass, the symbol-table is sorted

and printed out.

In the later chapters, the Cross-Assembler will be described in more detail. The following section contains a brief introduction to the Altair 680b organization.

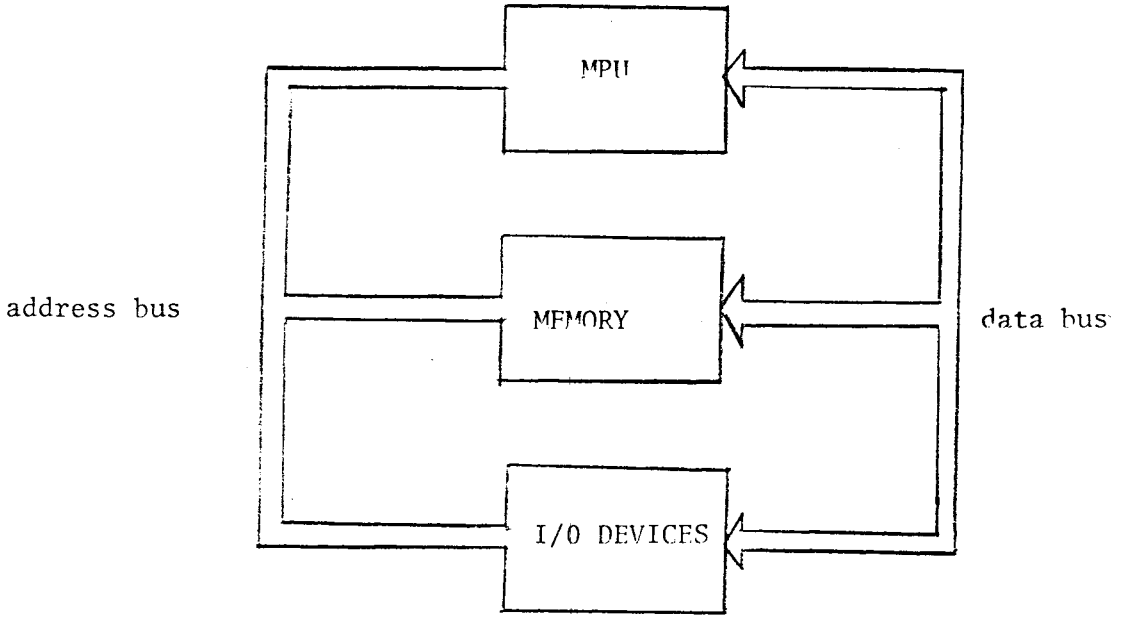
1.4 Altair 680b Micro-Computer (4, 3)

The computer consists of a micro-processing unit (MPU), a memory and a number of input and output devices. These components are linked together by an address bus and data bus. (see Fig. 1-1)

The computer memory is used to store instructions and data for use by the MPU. In the 680b, the memory is organized into 8-bit words, called bytes. Each memory byte is assigned an unique 16-bit address. This address is used by the MPU to gain access to the contents of a particular memory byte.

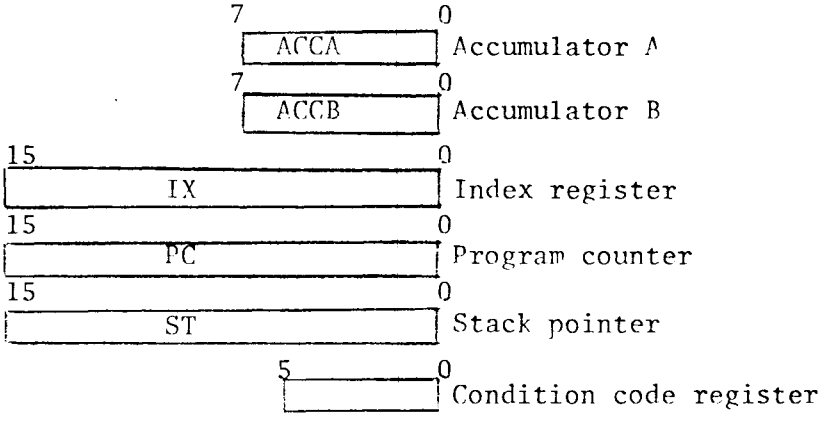
The MPU is a Motorola M6800, which operates on 8-bit binary numbers presented to it via the data bus. A given number (byte) may represent data or an instruction to be executed, depending on where it is encountered in the control program. The M6800 has 72 unique instructions. However, it recognizes and takes action on 197 of the 256 possibilities that can occur using an 8-bit word length. The larger number of instructions results from the fact that many of the executive instructions have more than one addressing mode.

The addressing modes refer to the manner in which the program causes the MPU to obtain instructions and data. The programmer must have a method for addressing the MPU's internal registers and all of the external memory locations. A programming model of the M6800 is



MICROCOMPUTER SYSTEM BLOCK DIAGRAM

Fig. 1-1



PROGRAMMING MODEL OF M6800

Fig. 1-2

shown in Fig. 1-2. The programmable registers consist of: 2 8-bit accumulators; a 6-bit condition code register; a program counter, a stack pointer and an index register, each 16-bits long.

The MPU operates in one of the following addressing modes:

Inherent

Immediate

Direct

Extended

Relative

Indexed

A number of instructions, either alone, or together with an accumulator operand, contain all of the address information required. Such instructions are said to be operating in the "inherent" mode of addressing.

In the Immediate addressing mode, the operand is the value that is to be operated on. No further address reference is required.

In the Direct and Extended modes of addressing, the operand field of the source statement is the address of the value that is to be operated on. The direct and extended modes differ only in the range of memory locations to which they can direct the MPU. Direct addressing generates a single 8-bit operand and can hence address only memory locations 0 through 255; a two byte operand is generated for extended addressing enabling the MPU to reach the remaining memory locations, 256 through 65535.

The Relative addressing mode, implemented for the MPU's branch instruction, specifies a memory location relative to the Program Counter's

current location. One byte is generated for the relative address. Since it is desirable to be able to branch in either direction, the 8-bit address byte is stored as a number in 8-bit, two's complement, binary form, with decimal value in the range from -128 to +127. This results in an effective range of -126 to +129 with respect to the branch instruction itself.

With Indexed addressing, the numerical address is variable and depends on the current contents of the Index-register. A byte is generated for the numerical value, which will be automatically added to the Index-register during execution.

The source statement format contains sufficient information for the selection of the addressing mode. For instructions which use both direct and extended modes, the Assembler selects the direct mode if the operand address is in the range 0 to 255 and the extended mode otherwise. There are a number of instructions for which the extended mode is valid but the direct is not. For these instructions the extended mode is selected, even if the operand address is in the range 0-255.

The assembler directives allow the programmer control of the assembly of the executive instructions into machine code. They provide for the allocation of memory, assignment of values to data, source listing format control and many other useful functions. A list of the assembler directives is given in Appendix-G.

1.5 Approach

The following points were kept in mind while designing and implementing the cross-assembler

(1) "Assembly Language is potentially the most dangerous of programming

languages, because of the variety with which a programmer can solve his problems." (5)

"Write clearly-don't be too clever".(6) Complex code has been kept to a minimum, and clarity and simplicity have been given (almost excessive) importance.

(2) Unconditional branching has been used with a great deal of caution. Except in transferring to the beginning of a loop, unconditional transfers "backwards" have been kept to a bare minimum.

(3) As far as possible, top-down design and testing techniques have been followed.

(4) "There is nothing in the programming field more despicable than an uncommented program" (5)

"...a more subtle problem can exist if the program is heavily laden with comments."(5)

As far as possible, comments have been included only wherever they are necessary.

CHAPTER II

THE ASSEMBLER

2.1 Introduction

This chapter discusses the overall top-level design of the Assembler. The various data-bases and structures used are defined, and the general algorithm for the assembly process is presented.

2.2 Functions

The basic functions which have to be performed by the Assembler are as follows:

(1) Generate object-code.

  (a) Generate machine instructions

  (b) Generate data words, allocate storage

(2) Produce source listing; symbol table.

Since symbolic addresses are permitted, the Assembler has to keep a note of the addresses of all the symbols used by the programmer. Because symbols can appear before they are defined, it is necessary to make two passes over the input. The first pass defines the symbols while the second pass produces the object code and source listing. The functional descriptions of the two passes are given below:

Pass-1:

(1) Obtain lengths of machine instructions.

(2) Allocate storage space.

(3) Maintain a location counter.

(4) Collect symbolic addresses and store them along with their addresses.

Pass-2:

(1) Assemble machine instructions using addresses of the symbols collected in pass-1.

(2) Generate data words.

(3) Generate source listing and symbol-table listing.

2.3 Data-structures

Having defined the functions of the Assembler, the next obvious step is the definition of the various data-structures. The major data-bases used are:

(1) Source-code line--INSLN

(2) Pseudo-operation table--POPTB

(3) Machine operation table--MOPTB

(4) Storage required by current instruction--LENTH

(5) Location-counter--LOCTR

(6) Symbol-table--SYMTB

(7) Error-Tables--ER1TB, ER2TB

(8) Direct-address-table--DIRAD

(9) Print-line--PRTLN

(10) Object-code buffer--BNOUT

INSLN contains the current source line. Since the NOVA has 16-bit words, 2 characters (7 or 8-bit ASCII) can be stored per word. Hence INSLN is a vector of 40 words (or 80 characters).

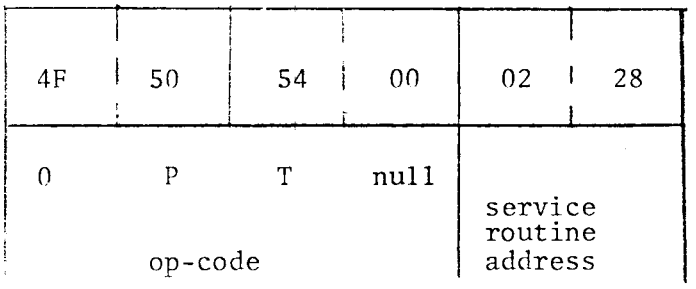
The source-line is checked to determine the type of opcode and operands. Two tables are used for this purpose--the pseudo-operation table and the machine-operation table.

The pseudo-op table contains a list of all pseudo-ops along with the corresponding service routine addresses. Each entry requires three 16-bit words. The first two words contain the pseudo-op mnemonic (3 characters), left justified, right filled with zeroes. The third word contains the service-routine address. A typical entry in the table is shown in Fig. 2-1.

The machine-operation table contains a list of all machine-operation code mnemonics. Each entry requires 3 words. The first two words contain the mnemonic, left justified and right filled with zeroes. The first byte of the third word contains the instruction type (numbered 0 to 8--see Appendix F) while the second byte contains the base value of the instruction. Depending on the mode of addressing used, the base value is modified to obtain the actual opcode. A typical entry is shown in Fig. 2-2.

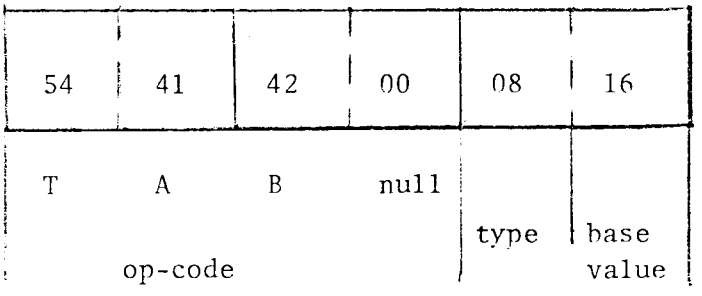
Once the instruction type has been obtained, the number of words required by the instruction is determined and LENTH is set accordingly. Before scanning the subsequent line, LOCTR is incremented by this amount.

If a label is present on a source line, it is stored in the symbol-table (SYMTB) along with the current location counter value. A label can have upto a maximum of six characters. Hence each entry in the table requires four words. The first three words contain the symbol, left justified, right filled with zeroes, while the fourth word contains



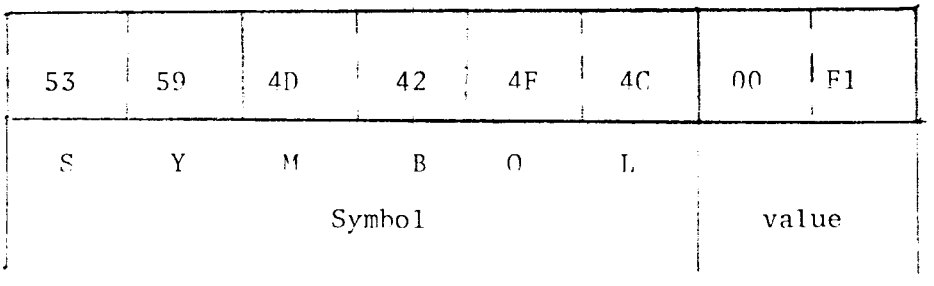
PSEUDO-OP TABLE ENTRY

Fig. 2-1



MACHINE-OP TABLE ENTRY

Fig. 2-2



SYMBOL TABLE ENTRY

Fig. 2-3

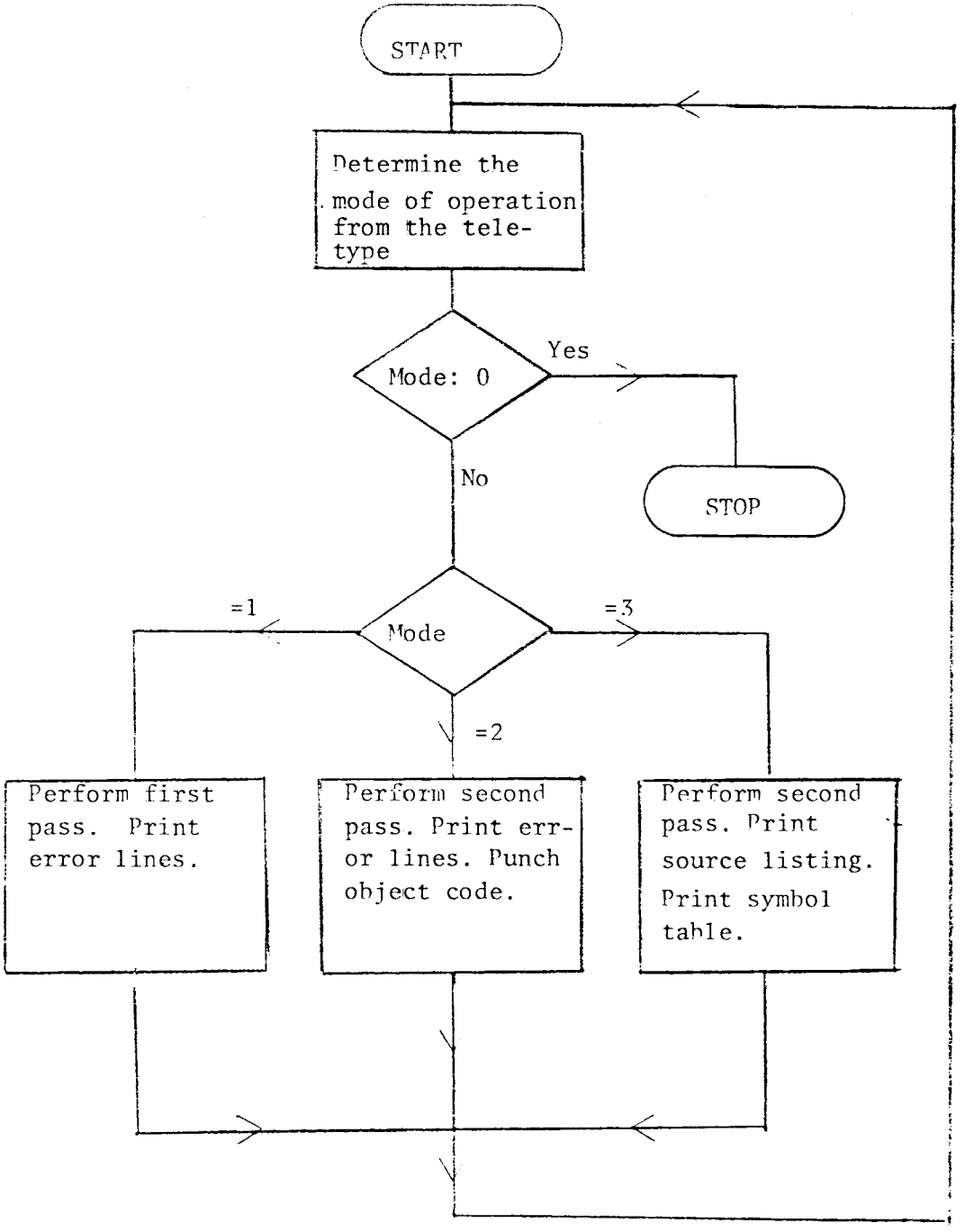
the corresponding location-counter value. The characters are represented by 7-bit ASCII strings. The format of a symbol-table entry is shown in Fig. 2-3.

The error table contains the line-number and the error-word of all of the error lines. A maximum of 25 entries can be accommodated. The direct-address table contains the line numbers of the instructions which use the direct mode of addressing. Such a table is essential, since the Assembler has to decide, in the first pass, whether certain instructions need the direct or extended mode of addressing. This table has space for upto 50 entries.

BNOUT and PRTLN are the object code buffer and the print-line buffer respectively. PRTLN can accommodate upto a maximum of 120 characters, and is flushed at the end of every scan. BNOUT is a vector of 40 words, and is punched only when a data-record (in the Motorola terminology, see Appendix C) has been assembled.

2.4 General Algorithm

The general flow of control is shown in Fig. 2-4. A more detailed description of the algorithm is given in the subsequent chapters.



GENERAL FLOW OF CONTROL

Fig. 2-4

CHAPTER III

FIRST PASS

3.1 Introduction

The general flow of logic was presented in Chapter 2. Here the first pass algorithm will be discussed in detail.

The source-code is processed line by line. The first character of each line is checked to determine whether the line is a comment. If so, no special action is taken. If a label is present, it is stored in the symbol table, along with the current value of the location counter. The operation code is identified, and necessary action is taken. The location counter is updated. The above steps are repeated until the END or MON pseudo-op is encountered. The detailed pass-1 flow chart is shown in Fig. 3-1.

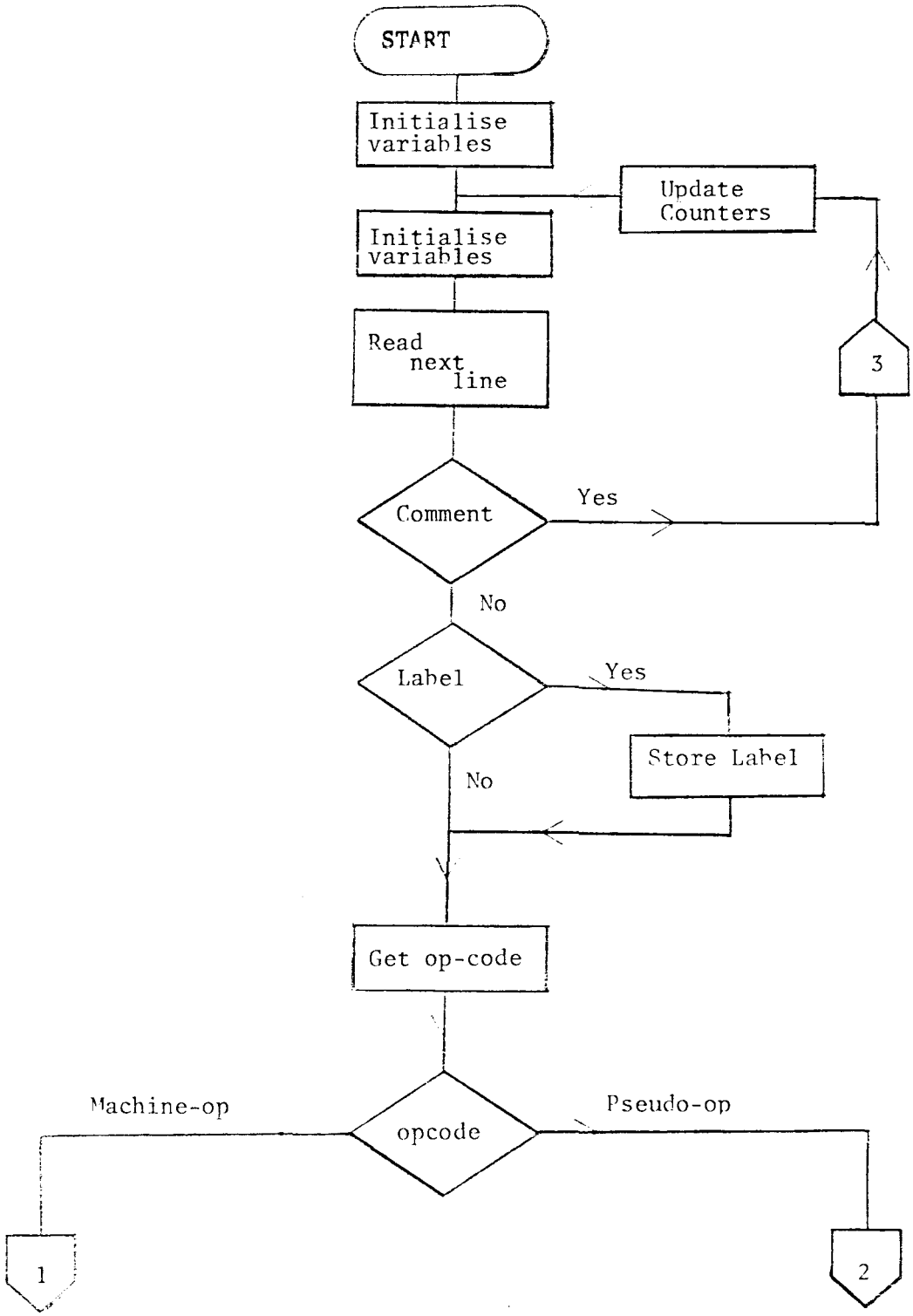
3.2 Machine-instruction processing

In the first pass, the machine-operation code service routines perform two major functions:

(1) Determine the length of the instructions

(2) Maintain the direct-address table.

To determine the length of an instruction, the addressing mode has to be determined. In most cases, an examination of the operands will identify the addressing mode. For instructions which can use either the direct or extended mode, the Assembler has to decide on the mode to be used. If the operand field cannot be evaluated, the



FIRST PASS

Fig. 3-1

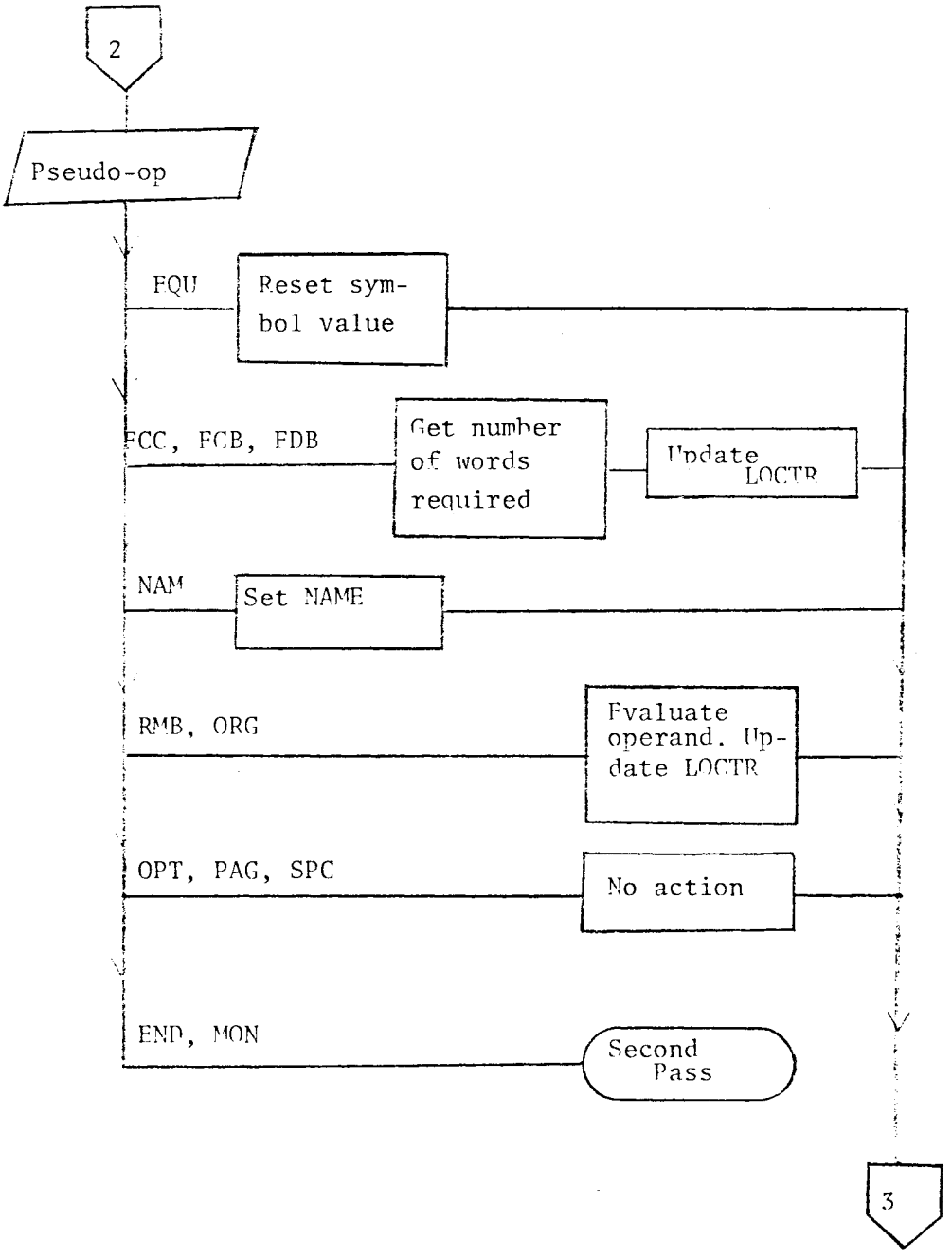

2
Pseudo-op
EQU
Reset sym-
bol value
FCC, FCB, FDB
Get number
of words
required
Update
LOCTR
NAM
Set NAME
RMB, ORG
Evaluate
operand. Up-
date LOCTR
OPT, PAG, SPC
No action
END, MON
Second
Pass
3

Fig. 3-1 (contd)

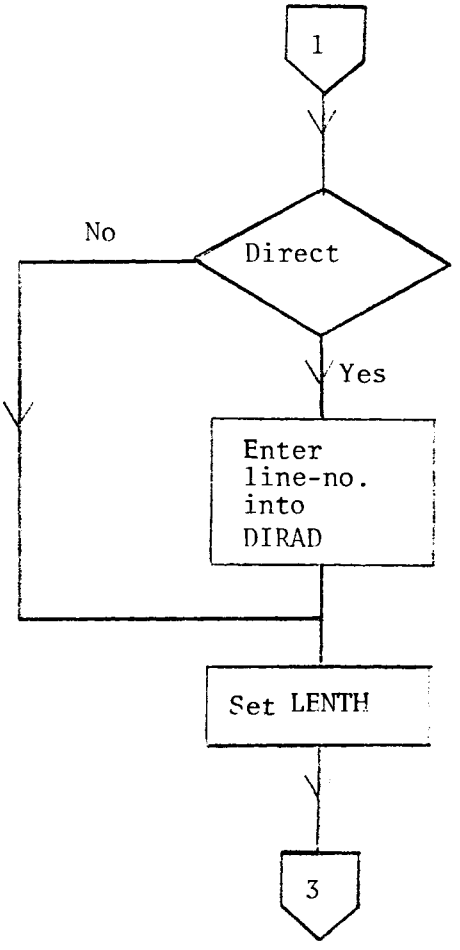


Fig. 3-1 (contd)

instruction is assumed to operate in the extended mode. If the operand field can be evaluated, the value is examined, and if it is less than 256, the direct mode is selected, and the line number of the instruction is entered in the direct-address table; otherwise the extended mode is used.

3.3 Pseudo-instruction Processing:

Basically, the Assembler recognizes about 12 pseudo-instructions (see Appendix G). Some of them result in machine-code, while others provide information regarding listing, storage requirements etc. to the Assembler. Pseudo-instruction processing in the first pass is described below.

END, MON:

These pseudo-ops signify the end of the program. The last line of the source-program should contain one of these instructions. There should be no symbol in the label field, while the characters after the pseudo-op are treated as comments. Pass-1 scanning is terminated and control returns back to the main-program.

EQU:

This pseudo-op defines the value of the symbol in the label-field. The label is initially entered, along with the current location counter value. When an EQU is recognized, the operand field is evaluated. If it can be evaluated, the symbol value is reset, otherwise it is deleted from the symbol table.

FCC, FDB, FCB:

These are constant defining pseudo-ops. They are processed to

determine the amount of storage required, and the location counter is updated accordingly.

NAM:

This pseudo-op defines the name of the program. Six characters after the pseudo-op are entered into "NAME", which is used in pass-2 while generating the source-listing and object code. The default name for the program is "START". Since this does not produce any machine code, the location counter is not incremented.

PAG, SPC:

These pseudo-ops are not processed in pass-1.

ORG:

This specifies the starting address of the following block of instructions/data. If the operand field can be evaluated, the location counter is reset. Otherwise the line is flagged as an error. The label field may not contain a label.

RMB:

This pseudo-op reserves a block of storage for data. If the operand field can be evaluated, the location counter is incremented accordingly; otherwise the line is flagged as an error.

3.4 Conclusion:

This concludes the detailed description of instruction processing in the first pass. At this point, all symbols (except some defined by EQU pseudo-ops) have been defined, and storage requirements established. The data-bases transmitted to pass-2 are:

(1) Direct address table

(2) Symbol table

(3) Error table

Control returns to the main program where the user is requested for further instructions via the tele-type. The second pass is described in the following chapter.

CHAPTER IV

SECOND PASS

4.1 Introduction

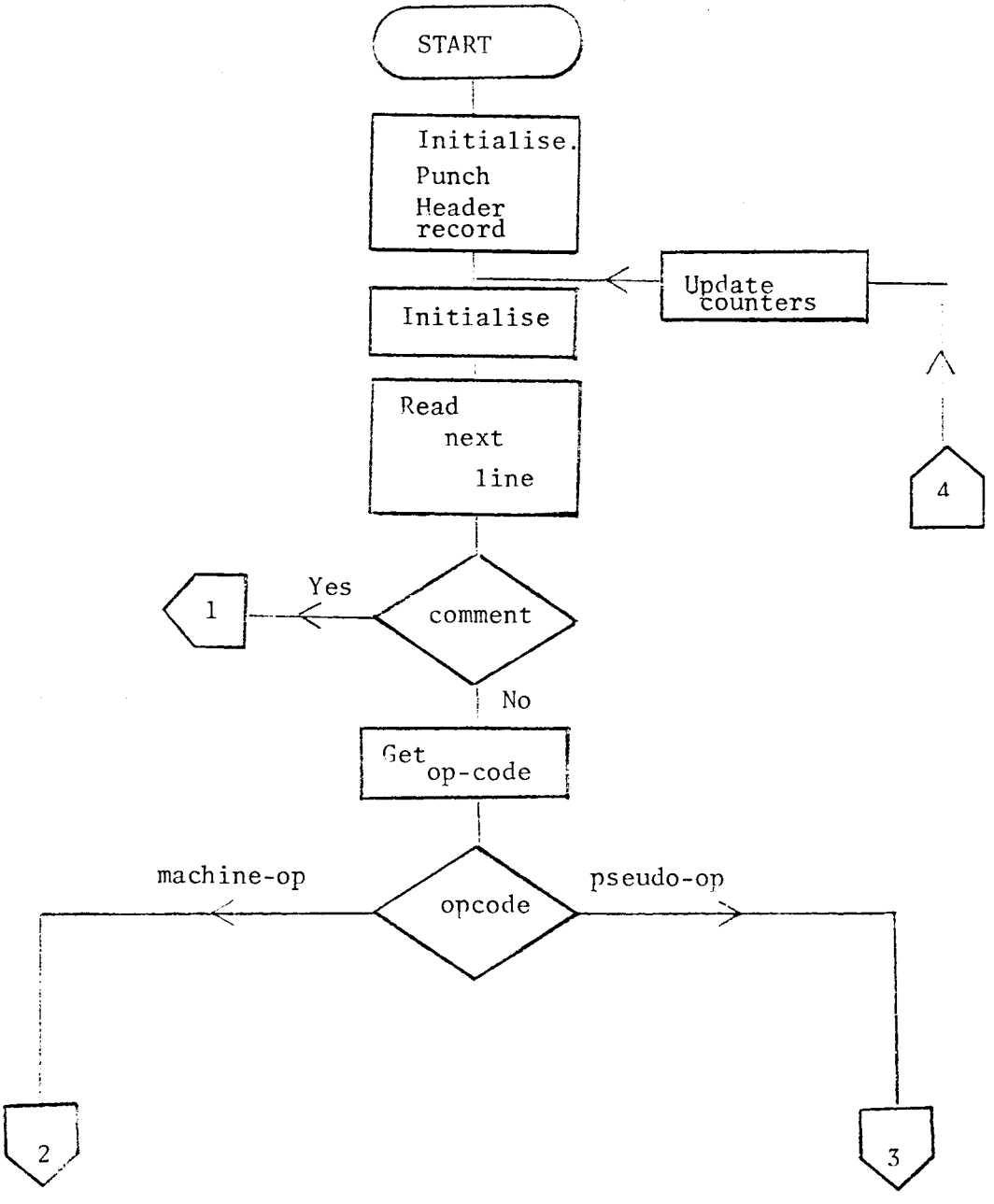
After all the symbols have been defined by pass-1, it is possible to finish the assembly by processing each line and determining the value of the opcode and its operand field. In addition, pass-2 must structure the generated object-code into a format suitable for acceptance by the microcomputer's resident PROM Monitor (See Appendix C), and produce a source-listing containing the original source line numbers, location and value of the bytes generated. A listing of the symbols and their values is also produced. Note that, depending on the mode requested by the user, either the source listing or the object code is generated by the Assembler. This is necessary since both the source-listing and the object-code have to be routed to the tele-type.

A location counter is maintained as in pass-1. The source code is read on a line by line basis. The opcode is examined, and control is transferred to the corresponding service routine.

4.2 Machine Instruction Processing

The major function of the machine instruction service routines is to assemble the instruction. For this the op-code has to be determined and the operand evaluated.

The base value of the opcode can be obtained from the Machine-



SECOND PASS
Fig. 4-1

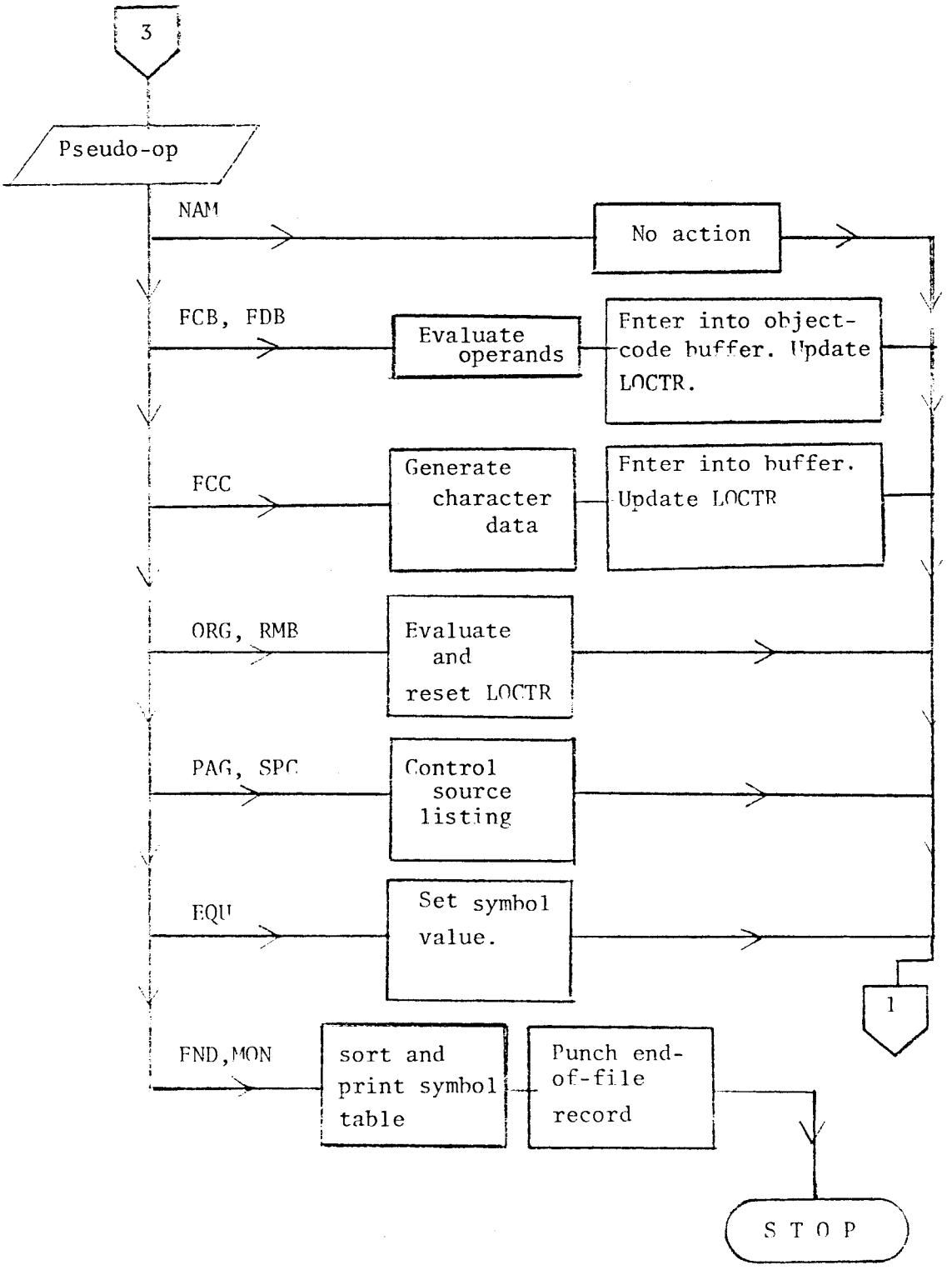

3
Pseudo-op
NAM
No action
FCB, FDB
Evaluate operands
Enter into object-code buffer. Update LOCTR.
FCC
Generate character data
Enter into buffer. Update LOCTR
ORG, RMB
Evaluate and reset LOCTR
PAG, SPC
Control source listing
EQU
Set symbol value.
1
END, MON
sort and print symbol table
Punch end-of-file record
STOP

Fig. 4-1 (contd)

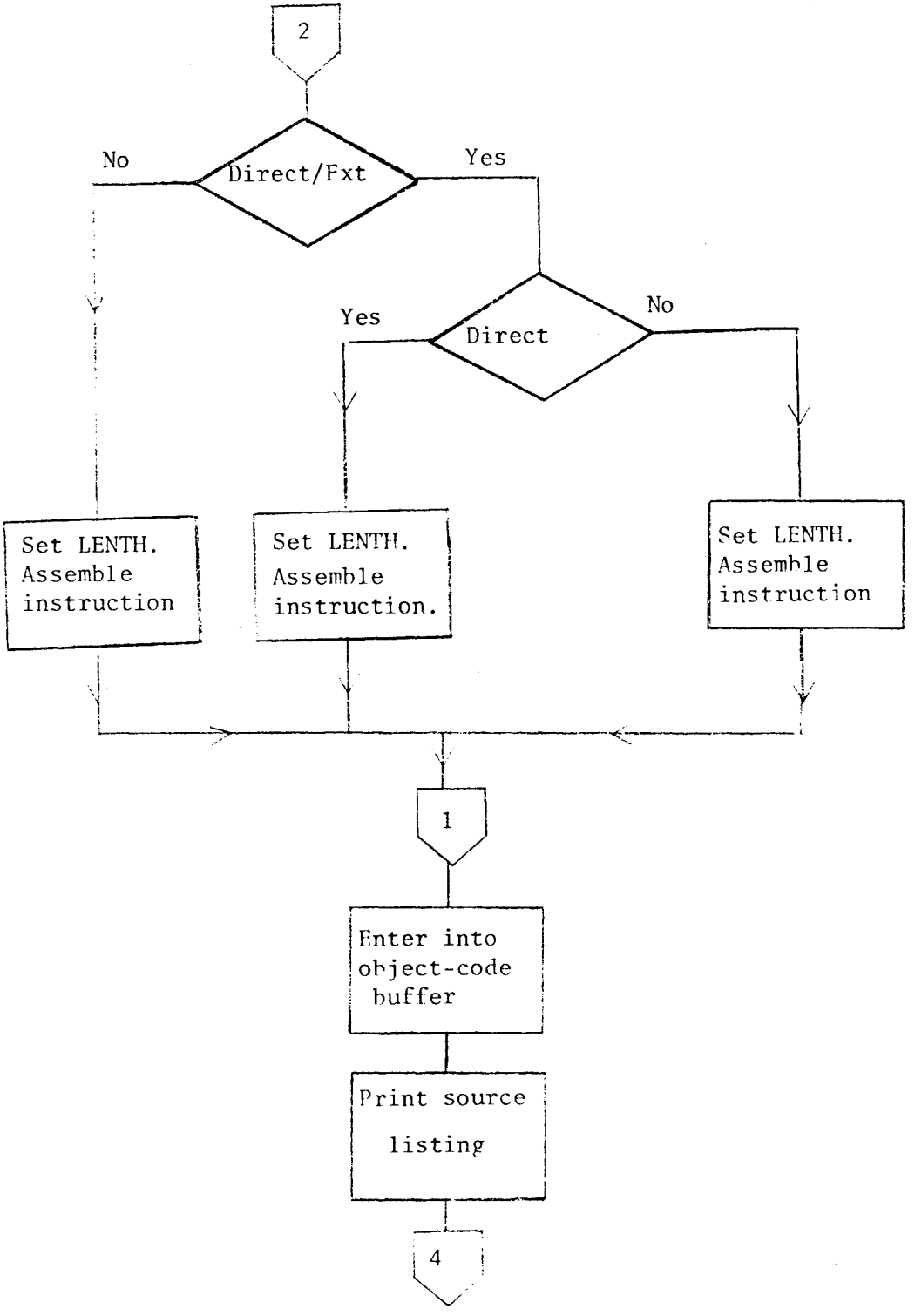

2
Direct/Fxt
No
Yes
Direct
Yes
No
Set LENTH.
Assemble
instruction
Set LENTH.
Assemble
instruction.
Set LENTH.
Assemble
instruction
1
Enter into
object-code
buffer
Print source
listing
4

Fig. 4-1 (contd)

operation table. Depending on the mode of addressing, the actual value can be evaluated. A scan of the operands will establish the mode, except in the case of instructions which can use either the direct or the extended mode of addressing. For this case, the direct address table is first searched to determine whether an entry corresponding to the line is present. If so, the mode of addressing to be used is direct; otherwise the extended mode is used.

The operand is evaluated using the symbol table produced by pass-1. If the instruction requires an 8-bit value, the least significant 8-bits are used. For the relative mode of addressing, the location counter value is used to obtain a relative address. The generated bytes are stored in WSPCE and LENTH is set to the number of bytes of generated code.

4.3 Pseudo-instruction processing

The actions taken for the various pseudo-ops are explained below.

END, MON:

These pseudo-ops signal that there are no more instructions to be processed. Pass-2 now proceeds to do the cleaning-up process.

EQU:

This pseudo-op needs processing in pass-2, since symbols which have not been defined before the instruction can appear in the operand field . If the operand field can be evaluated, the value associated with the label is reset; otherwise, the label is deleted from the symbol-table.

FCB, FDB:

These pseudo-ops generate constant data-bytes. FCB generates an 8-bit byte while FDB generates two 8-bit bytes of data. Since more than one constant can be defined per line, each of the operands is evaluated,

entered into the object-code buffer, and the location counter is updated. If any of the operands cannot be evaluated, the line is flagged as an error.

FCC:

This pseudo-op generates 7-bit ASCII character constants. Basically, there are two ways of defining character strings--by using delimiters or by defining the length. The Assembler generates the 7-bit ASCII equivalents, enters them into the object-code buffer and updates the location-counter.

ORG, RMB:

Processing is identical to that of pass-1.

PAG, SPC:

These pseudo-ops control the assembly source-listing. For the PAG pseudo-ops, the tele-type is positioned at the top of a new page, and the title lines are printed. For the SPC, the operand field is evaluated, and a corresponding number of blank lines are printed. Neither of the pseudo-ops is printed out.

NAM:

No action is necessary in pass-2.

4.4 Conclusion

At the end of each scan, the source-line is printed out. Any machine code generated is entered into the object-code buffer. When the END/MON pseudo-op is encountered, source-line scanning is stopped. The total number of errors is printed. The object-code buffer is flushed, and an end-of-file record is punched. The symbol table is sorted and

listed, and control returns back to the main program.

CHAPTER V

TABLE HANDLING

5.1 Introduction

This chapter describes the handling of the various tables used by the Assembler. The tables discussed are

(1) Symbol-table

(2) Machine-operation-code table.

All the other tables use a linear search technique.

5.2 Symbol-table-Searching

The symbol table is hash-coded. In order to use a hash-coded table, two decisions have to be made:

(1) Choose a hashing function

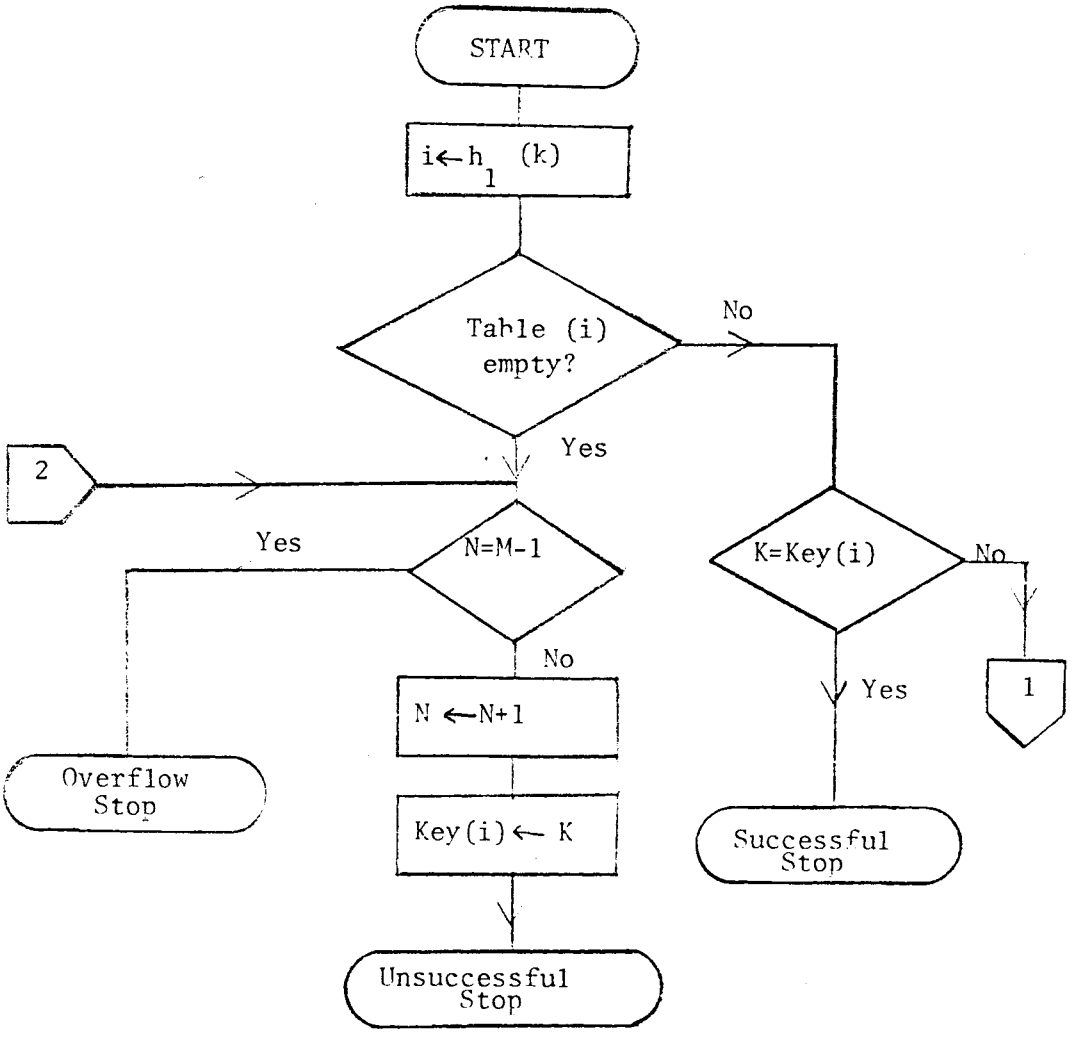
(2) Select a method for handling collisions

The symbol table uses a multiplicative hashing function (pp 508-9; ref 8). If w is the word size of the computer, A is any number relatively prime to w, and $0 \leq h(k) < M = 2^m$, the hashing function,

$$h(k) = \lfloor M((A/w \times k) \bmod 1) \rfloor$$

gives a value between 0 and M-1. Here h(k) consists of the m leading bits of the least significant $\log_2 w$ bits of the product AK.

Collision resolution is done by open-addressing. The idea is to "formulate some rule by which every key K determines a 'probe sequence', namely a sequence of table positions which are to be inspected whenever K



SYMBOL TABLE SEARCHING

Fig. 5-1

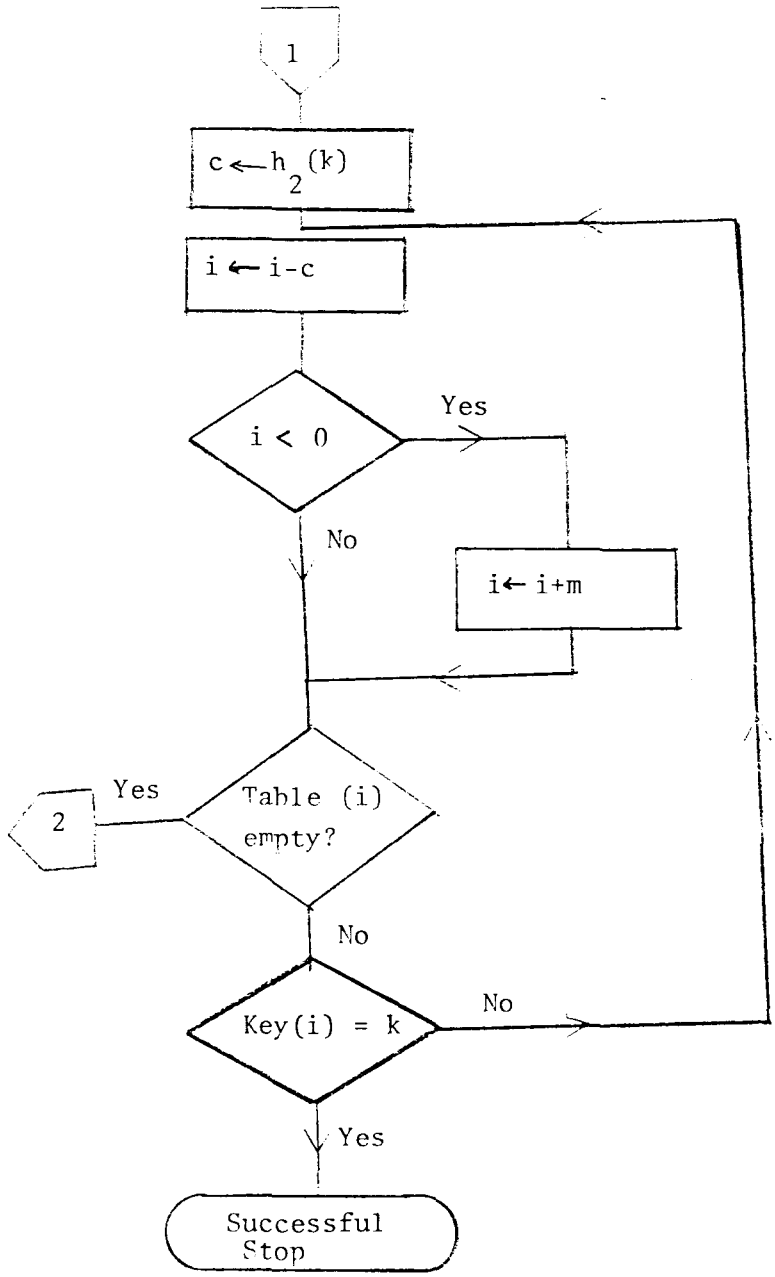

1
c ← h2(k)
i ← i-c
i < 0
Yes
No
i ← i+m
Table (i) empty?
Yes
2
No
Key(i) = k
No
Yes
Successful Stop

Fig 5-1 (contd)

is inserted or looked up" (pp 518, ref. 8). The method used in the Assembler uses 2 hash functions, $h_1(K)$ and $h_2(K)$. $h_1(K)$ produces a value between 0 and M-1, while $h_2(K)$ produces a value between 1 and M-1, relatively prime to M. If $M = 2^m$, $h_2(K)$ can be obtained by shifting AK mod w m more bits to the left and orring in a 1.

The flow chart for the algorithm is given in Fig. 5-1. (N denotes the number of entries in the symbol table).

5.3 Symbol-table Sorting

The sorting algorithm used here is based on two things:

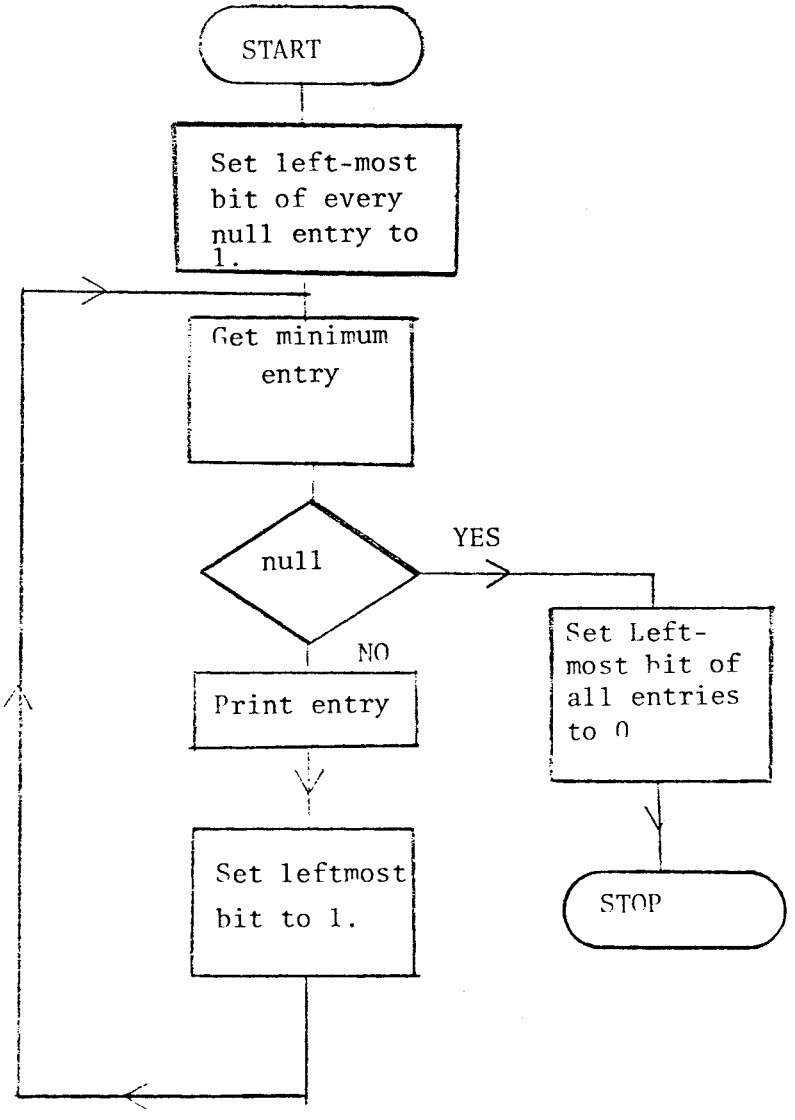
(1) The first bit of all entries is always 0, since 7-bit ASCII representation is used.

(2) The sorting mechanism need not be fast, since the machine will be working in a single user environment, with the Tele-type operated in the SKIP mode.

Initially, the first bit of all entries in the table is 0 . An initial pass is made through the table, and the left most bit of every null entry is set to 1. Thereafter, until the minimum entry in the table is a null with a 1 in the left most bit, the minimum entry is determined, printed out, and a 1 entered in the most significant bit. After all entries have been printed out, one more pass is done in order to reset the left-most bit of every entry in the table to 0 . The flow-chart for the algorithm is shown in Fig. 5-2.

5.4 Machine-operation table searching

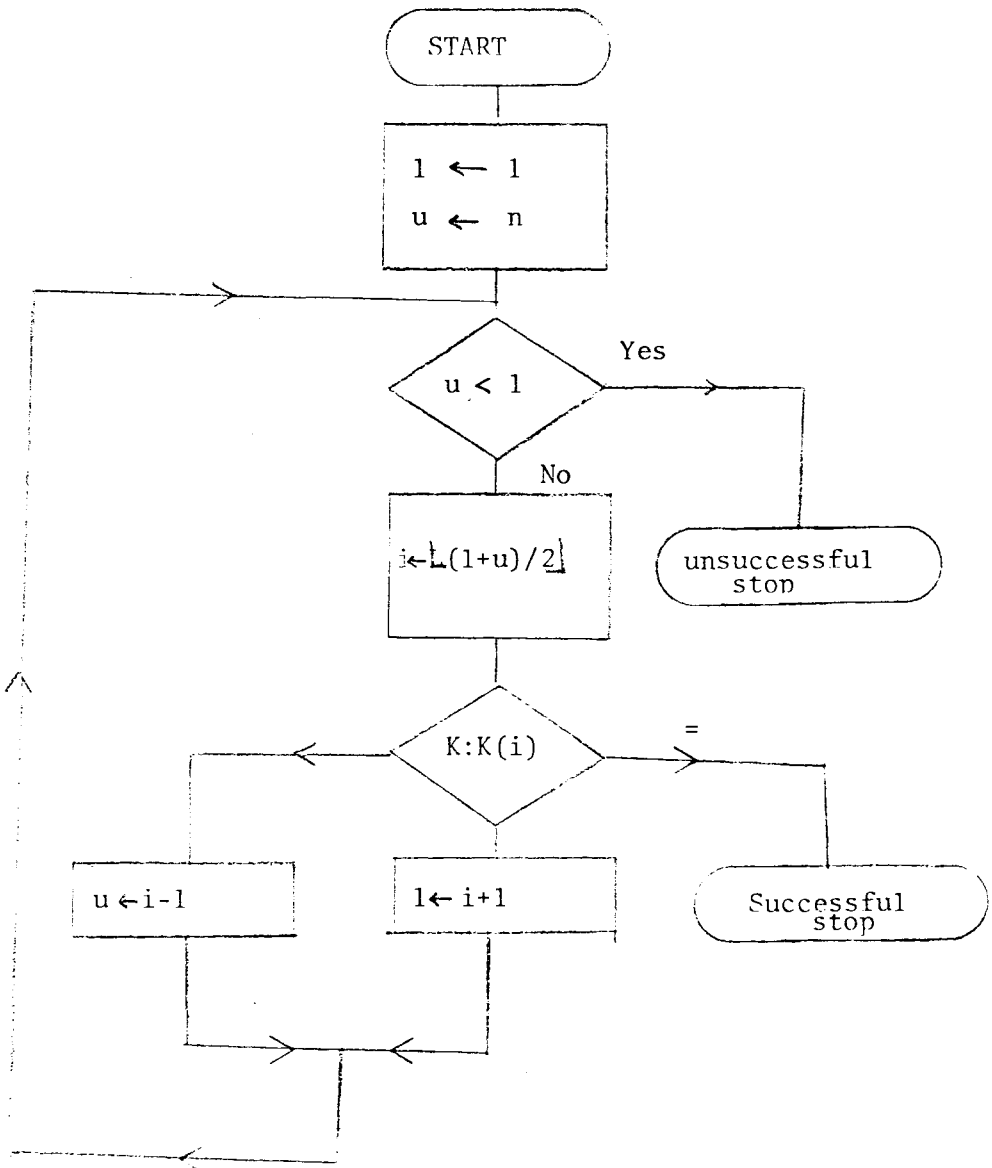
The machine operation table is a static table, with no insertions or deletions. It can be prearranged lexicographically, and hence a



SYMBOL TABLE SORTING

Fig. 5-2

binary search technique is used. The flow chart for the algorithm (pp. 407-408; ref. 8) is given in Fig. 5-3.



MACHINE OPERATION TABLE SEARCHING

FIG 5-3

CHAPTER VI

CONCLUSION

The author regrets the lack of many useful features in the Assembler, such as the OPT pseudo-op, cross-reference table, more detailed error-processing, better source-listing format control etc. These features could not be implemented due to lack of time.

The author found that modular design was very useful in writing software, especially in minimising and locating bugs during testing. Early in the implementation stage, the author learnt the hard way that proper documentation helps a great deal in debugging. The author also feels that proper modular design is more important than structured coding, since any un-structured code present is localised to a module.

Since the project was primarily an exercise in software design and development, the author feels that he has achieved what he had set about to do--get an idea of the problems involved in the design and development of software.

APPENDIX - A

INSTRUCTIONS TO USER

The starting address of the program is 2000_8. When started at this location, the "MODE :=" request is printed out.

The Assembler operates in 4 modes.

Mode 0:- The Assembler halts processing

Mode 1:- Perform pass-1 processing.

Print all error lines.

Mode 2:- Perform pass-2

Print error-lines

Punch object code.

Mode 3:- Perform pass-2

Generate source-listing

List Symbol table.

The Assembler occupies locations

$(40)_8$ thro' $(131)_8$

$(200)_8$ thro' $(341)_8$

$(2000)_8$ thro' $(11464)_8$

$(12000)_8$ thro' $(15060)_8$

APPENDIX - B

ERROR MESSAGES

CODE	MEANING
D	Delimiter not found in FCC
E	Illegal expression
F	Format error
I	Invalid Label
L	String too long in FCC
M	Multiply defined label
N	Label not present in EQU
O	Opcode unrecognizable
P	Label present in ORG, END
R	Expression cannot be evaluated in pass-1, ORG, RMB
S	Symbol too long
U	Undefined Symbol
V	Value overflow, division by 0.

PROGRAMS WITH ERRORS

```
ERR1                                                    PAGE  00001
 ER. LINE. LOC.  VALUE.   INPUT.

     00001                    NAM    ERR1
     00002              *  ERROR-CODES TEST.
     00003              *
     00004              *
     00005              ***** LABEL NOT PRESENT ,CODE-N *****
     00006              *
     00007        00FF  AAA    EQU    $FF
 N   00008        0000        EQU    $FF
     00009              *
     00010              ***** FORMAT ERROR,CODE-F *****
     00011              *
 F   00012 0000              ADCX    #$10
 F   00013 0000              ADC     #$10
 F   00014 0000  9900        ADC A
 F   00015 0002  9700        STA A
 F   00016 0004              STAX    $10
 F   00017 0004              ASLX
 F   00018 0004  00          PSH
 F   00019 0005  00          PSH C
 F   00020 0006              CPXA    #$10
```

```
ERR1                                                    PAGE  00002
 ER. LINE. LOC.  VALUE.   INPUT.

 F   00021 0006                    JMPA        $10
 F   00022 0006                    BCCA        $10
 F   00023 0006          AAK
 F   00024 0006
     00025               *
     00026               ***** EXPRESSION CANNOT BE EVALUATED
     00027               *      IN PASS-1,CODE-R ******
     00028               *
 R   00029 0006  0200              RMB         SSS
 R   00030 0200                    ORG         SSS
     00031         0200  SSS       EQU         $200
     00032               *
     00033               ***** LABEL PRESENT,CODE-P *****
     00034               *
 P   00035 0200          MNO       ORG         $300
 PM  00036               MNO       END
 TOTAL ERRORS  00018
```

```
ERR2                                                    PAGE  00001

 ER. LINE. LOC.  VALUE.   INPUT.
     00001                        NAM    ERR2
     00002              *  ERROR-CODES TEST.
     00003              *
     00004              *
     00005              ***** INVALID LABEL,CODE-I *****
     00006              *
 I   00007 0000  16     A            TAB
 I   00008 0001  16     B            TAB
 I   00009 0002  16     X            TAB
 I   00010 0003  16     1AC          TAB
 I   00011 0004  16     ABCDEFG      TAB
 I   00012 0005  16     AB$          TAB
     00013              *
     00014              ***** MULTIPLY DEFINED LABEL,CODE-M *****
     00015              *
     00016 0006  16     KKK          TAB
 M   00017 0007  16     KKK          TAB
     00018              *
     00019              ***** DELIMITER NOT FOUND,CODE-D *****
     00020              *
```

```
ERR2                                                    PAGE  00002
 ER. LINE. LOC.  VALUE.                 INPUT.

     00021 0008                                 FCC     /TEXT/
 D   00022 000C                                 FCC     /TEXT
     00023                          *
     00024                          ***** STRING TOO LONG,
                                           CODE-L *****
     00025                          *
 L   00026 004A                                 FCC     256
     00027 004A                                 FCC     255,
     00028                          *
     00029                          ***** OPCODE UNRECOGNIZABLE,
                                           CODE-O *****
     00030                          *
     00031 0149  06                             TAP
 O   00032 014A                     TT          ABC
     00033                          *
     00034                          ***** UNDEFINED SYMBOL,
                                           CODE-U *****
     00035                          *
 U   00036 014A  27B4                           BEQ     CDE
     00037 014C  27B2                           BEQ     CED*2
 U   00038 014E  24B0                           BCC X
 U   00039 0150  FF00 00            *           STX A   $10
 U   00040 0153  7E00 00                        JMP     #$10
```

```
ERR2                                                    PAGE  00003
 ER. LINE. LOC.  VALUE.   INPUT.

 U   00041 0156  27A8          BEQ       $10*-$2
     00042             *
     00043             ***** SYMBOL TOO LONG,CODE-S *****
     00044             *
 S   00045 0158  27A6          BEQ       ABCDEFG
     00046             *
     00047             ***** ILLEGAL EXPRESSION,CODE-E *****
     00048             *
 E   00049 015A  27A4          BEQ       $10GH*$2
 E   00050 015C  27A2          BEQ       %112*$3
     00051             *
     00052             ***** VALUE OVERFLOW,CODE-V *****
     00053             *
 V   00054 015E  27A0          BEQ       $10/0
 V   00055 0160  279E          BEQ       $10/$0
     00056                 END
 TOTAL ERRORS  00021
```

APPENDIX - C

ABSOLUTE BINARY FORMAT

The PROM monitor supports the paper-tape format established by Motorola.

The first character of a record is an S. The digit following the S defines the type of record.

S_0 = Header Record

S1 = Data Record

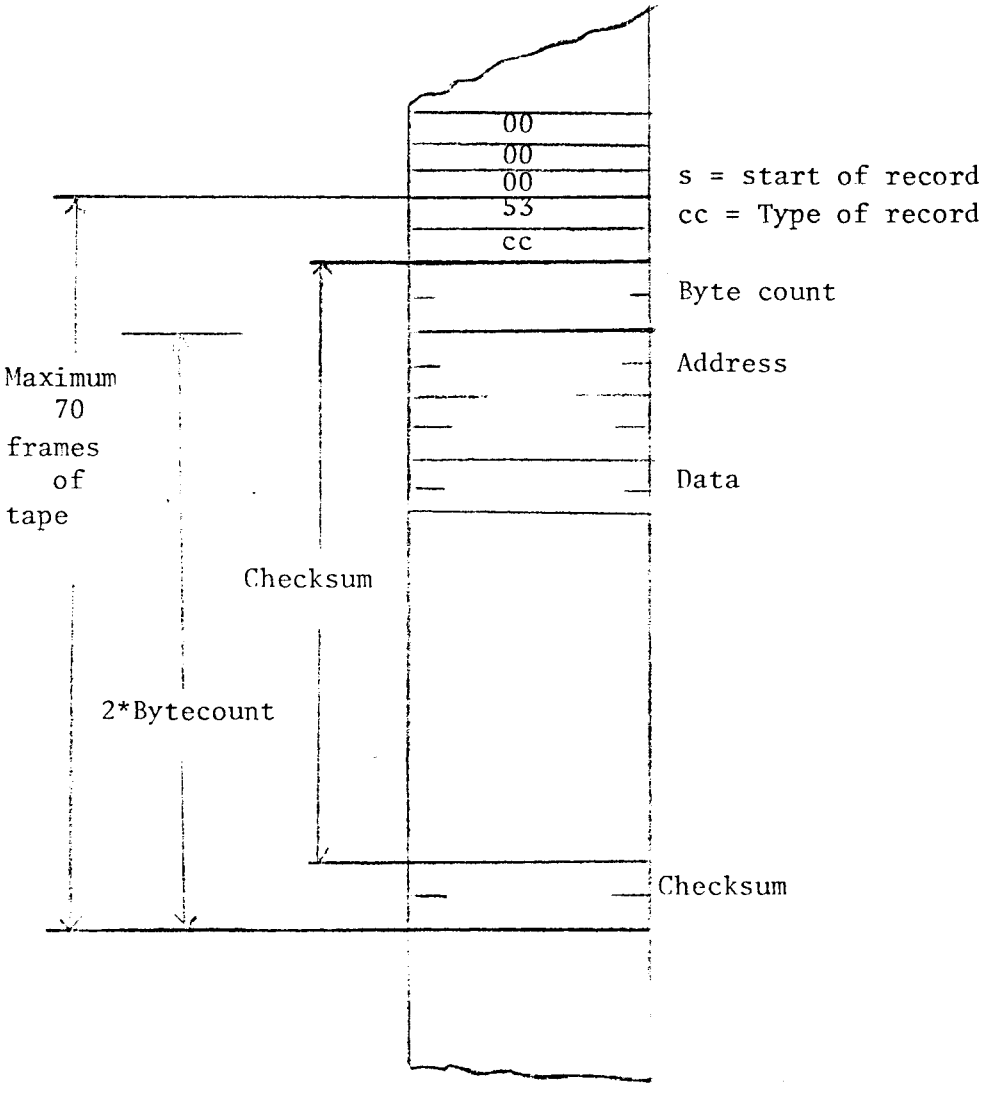
S9 = End of File Record

Header records (type S0) contain the program name, and are ignored by the PROM Monitor. The end of file record (type S9) causes the monitor to terminate the loading process. Data records (type S1) contain the actual data to be loaded and are of the form:

S1 NNAAAADD.....DDCC

where S1 specifies the record is a data-record, NN is a two digit hexadecimal byte count specifying the number of remaining bytes in the record (1 byte = 2 frames of tape), AAAA is the 4 digit hexadecimal starting address of the data-block, each DD pair consists of two hexadecimal digits which are combined to form a byte, and CC is the checksum of all preceding frames (excluding S and 1). The checksum is the 1's complement of the binary sum of the byte count, the address, and the data bytes.

Further information concerning the paper tape is given in Fig. C-1.



MOTOROLA PAPER TAPE FORMAT

Fig. C-1

APPENDIX - D

I-O INTERFACE

The various I-0 devices used by the Assembler are:

(1) BPNDV - Binary punch device. Object code is routed to this device

(2) ECHODV - Echo device. Input characters are echoed onto this device

(3) MDIND - mode input device. Reply to the "Mode=" request is received from this device

(4) MDOTD - "Mode=" message is printed on this device

(5) SCIND - Souce code input device. Source code is accepted from this device

(6) SLOTD - Source Listing is printed on this device

The above are located in locations $(40)_8$ through $(45)_8$ of the Assembler. The default values are $(10)_8$ for all input devices and $(11)_8$ for all output devices. Once the Assembler has been loaded, the user can set these to any desired device-code.

The basic input and output routines are PUT75 and GET78. GET78 gets the next character from the device whose device code is in "INPDV" into accumulator 0 while PUT75 transmits the character in accumulator 0 to the device whose device code is in OUTDV. Any I-0 changes can be implemented by modifying these 2 routines appropriately.

APPENDIX - E

LANGUAGE DESCRIPTION

E-1 Character Set

(1) The alphabet A-Z, integers 0-9 arithmetic operators

+ - * /

(2) Special prefix characters

specifies immediate mode of addressing

$ specifies a hexadecimal number

@ specifies an octal number

% specifies a binary number

' specifies an ASCII character

(3) Separating characters. SPACE, COMMA and CR(Carriage Return)

(4) A comment in the source statement may include any character except the CR and LF (line feed)

(5) In addition to the above, the Assembler has the capability of reading strings of characters and of entering the corresponding 7-bit ASCII code into specified locations in memory. This capability is provided by the Assembler directive FCC.

E-2 Fields of the source statement

A source statement includes from one to four fields. From left to right, the 4 fields are: (1) Label (2) Operator (3) Operand (4) Comment. The comment is optional, and may be used in all source

statements. A label is required for some statements which are involved in the definition of symbols and, in some cases, at the destinations of branches and jump instructions. An operand field may or may not be present depending on the nature of the operator. The mnemonic operator must be present in every statement except when the whole line is a comment. In that case, the first character in the source line should be an * .

With one exception, the successive fields within the statement are separated by one or more SPACE characters. The exception applies to instructions which have an accumulator as an operand. The programmer has the option of including the accumulator character (A or B) along with the op-code mnemonic, resulting in an apparent 4 character format, as for example, "ADCA", "ASRB", "STAA" etc.

A label, if used, must begin in the first character position of the source-statement. A space in the first character position is used to indicate that a label is not included in the statement. Except in some cases when it is used with the mnemonic operator EQU, a label always corresponds to a numerical address. A label consists of 1 to 6 alpha numeric characters, with the first character alphabetic. A label should not consist of anyone of the single characters A, B or X. All labels within a program should be unique.

The mnemonic operators recognized by the Assembler include 72 executable instructions. Each instruction is translated into one to three bytes of machine code. The remaining mnemonic operators are assembler directives of which 3 (FCB, FDB, FCC) are translated into

one or more bytes of machine code.

The kind of information placed in the operand field depends on the particular mnemonic operator. For the 72 executable instructions, the microprocessor uses various modes of addressing for obtaining the operand address. The addressing mode is determined by the mnemonic operator combined with the information in the operand field. The Assembler recognizes numbers, symbols and expressions in the operand field. Numbers are accepted by the assembler in the following formats:

Number (decimal)

$ Number (hexadecimal)

@ Number (octal)

% Number (binary)

where Number is a positive integer.

Symbols, when used in the operand field, must have been defined by appearing in the label field of any of the executable instructions, or the pseudo-ops FCC, FDB, FCC and RMB. The special symbol * represents the program counter. Expressions are combinations of symbols and/or numbers being separated one from the next by one of the arithmetic operators (+, -, * or /). The expressions are evaluated from left to right without any hierarchy of precedence among the operators.

APPENDIX F

GROUPED LISTING

The grouped listing of the 72 machine instructions is given below.

(For further information regarding this, refer to pp. A70-A71, ref. 3)

Type 0:	ADC	ADD	AND	BIT	CMP	
	EOR	LDA	ORA	SBC	SUB	
Type 1:	STA					
Type 2:	ASL	ASR	CLR	COM	DEC	INC
	LSR	NEG	ROL	ROR	TST	
Type 3:	PSH	PUL				
Type 4:	CPX	LDX	LDS			
Type 5:	STS	STX				
Type 6:	JMP	JSR				
Type 7:	BCC	BCS	BEQ	BGE	BGT	
	BHI	BLE	BLS	BLT	BHI	
	BNE	BPL	BRA	BSR	BVC	
	BVS					
Type 8:	ABA	CBA	DAA	SBA	TAB	TBA
	DEX	DES	INX	INS	TXS	TSX
	NOP	RTI	RTS	SWI	WAI	CLC
	CLI	CLV	SEC	SEI	SEV	TAP
	TPA.					

APPENDIX - G

PSEUDO-INSTRUCTIONS

END - End of program: When the assembler directive "END" is used, it marks the end of the source program. No other statement may follow the END directive. The operator in the last statement of the source program must be either MON or END. The END directive must not be written with a label, and it does not have an operand. It is not translated into object code.

EQU - Equate symbol: This is used to assign a value to a symbol. The EQU statement must contain a label, which is identical to the symbol being defined. The operand field may contain an expression which can be evaluated by the Assembler. The statement is not translated into object code.

FCB - Form constant byte: This directive may have one or more operands, separated by commas. An 8-bit unsigned binary number corresponding to each operand is in a byte of the object program. If there is more than one operand, they are stored in successive bytes. The operand field may contain any expression that can be evaluated by the assembler. The directive may be written, with a label. An FCB directive followed by one or more void operands separated by commas will store zeroes for the void operands.

FCC - Form Constant characters: This translates strings of characters into their 7-bit ASCII codes. Any character other than CR and LF can

processed by this directive.

(1) Count, comma, text, where the count specifies how many ASCII characters to generate and the text begins following the first comma of the operand. Should the count be longer than the text, spaces will be inserted to fill the count. Maximum count is 255.

(2) Text enclosed between identical delimiters, each being a single character. (If the delimiters are numbers, the text must not begin with a comma.)

The FCC directive may be written with a label.

FDB - Form double constant byte: This directive is identical to the FCB directive, except that a 16-bit unsigned binary number is stored corresponding to the values of each operand.

MON - Return to monitor: This directive is handled in an identical manner to the END directive.

NAM - Name: Provides a name for the program. The first six characters in the operand field are used. This is used in the top-of-the-page heading, and in the header-record of the object-code file. No object code results from NAM.

OPT - Option: This directive is ignored by the Assembler. No action is taken.

ORG - Origin: This defines the numerical address of the first byte of machine code resulting from the assembly of the immediately subsequent section of the source-program. The program counter is set to the value of the expression in the operand field. The ORG directive should not

have a label and does not translate into object-code.

RMB - Reserve Memory bytes: This causes the location counter to be increased by the value of the expression in the operant field. This reserves a block of memory whose length (in no. of words) is equal to the value of the operand field. The block of memory is changed. A label may be present on the statement.

SPC - Space n lines: This statement does not appear in the listing. The operand field is evaluated, and a corresponding number of lines are left blank.

PAG - Advance to the top of next page: This statement does not appear in the listing. This causes the Assembler to advance the paper to the top of the next page.

APPENDIX H

Program Listing

SUBROUTINE-INDEX

```
;*****************************************************
;
;   ***   6800 MICRO-PROCESSOR ASSEMBLER  ***
;   ***   WRITTEN BY:      P.S.KU AM
;   ***   DATE:            DEC. 1977
;
;*****************************************************
;
; THIS PROGRAM ASSEMBLES THE 6800 MICRO-PROCESSOR
; SYMBOLIC CODE AS DEFINED IN THE REPORT,AND PRODUCES
; A LISTING AND OBJECT CODE. THE OBJECT CODE IS IN
; PAPER-TAPE FORMAT,AS DEFINED BY MOTOROLA.
;
;*****************************************************
;
        .LOC    40
;
;*****************************************************
;
; THE FOLLOWING GLOBAL VARIABLES ARE USED IN THE
; PROGRAM. THESE ARE PLACED IN PAGE ZERO FOR EASY
; ACCESS BY VARIOUS ROUTINES.
;
;*****************************************************
;
;*****************************************************
;
; THE SIX  GLOBAL VARIABLES DEFINED BELOW REPRESENT
; THE VARIOUS DEVICE CODES.
;
;*****************************************************
;
PNCV:           11                      ;BINARY PUNCH
;                                       ; OUTPUT DEVICE.
ECOCV:          0                       ;ECHO DEVICE
;                                       CODE.
MCINC:          10                      ; CODE QUERY
;                                         INPUT DEVICE.
;
```

```
MQCTO:                 11                    ; MODE QUERY
;                                            ; OUTPUT DEVICE.
SCINO:                 10                    ; SOURCE CODE
;                                            INPUT DEVICE.
SLCTO:                 11                    ; SOURCE LISTING
;                                            ; OUTPUT DEVICE.
;
;****************************************************************
;
;
;
ABFLG:       .BLK      1                     ; THIS FLAG IS
;                                            ; USED TO FIND
;                                            ; OUT WHETHER
;                                            ; THE INST. HAS
;                                            ; ADD A,B OR
;                                            ; NEITHER.
ADLBL:       .BLK      1                     ; ADRESS OF CURR
;                                            ; -ENT LABEL.
BNFLG:       .BLK      1                     ; WHETHER OBJECT
;                                            ; CODE HAS TO BE
;                                            ; PUNCHED.
BNOUT:                 BNOU1                 ; ADDRESS OF
;                                            OBJECT CODE
;                                            BUFFER.
BREAK:       .BLK       1                    ; CONTAINS THE
;                                            CURRENT BREAK
;                                            CHARACTER.
BSVAL:       .BLK      1                     ;BASE VALUE OF
;                                            CURRENT INST.
CHPTR:       .BLK      1                     ;POINTS TO NEXT
;                                            POSITION IN
;                                            INPUT LINE.
CMBEG:       .BLK      1                     ;STARTING ADD.
;                                            OF COMMENTS IN
;                                            INPUT LINE.
CMFLG:       .BLK      1                     ;FLAG DENOTING
;                                            WHETHER COMMENT.
DEFNM:                 DEFN1                 ; DEF. NAME ADD.
DIRAD:                 DIRA1                 ; ADDRESS OF
;                                            ≠DIRAD≠.
DREPT:       .BLK      1                     ;  LAST ENTRY IN
;                                            ≠DIRAD≠.
E1EPT:       .BLK      1                     ; LAST ENTRY OF
;                                            ≠ER1TB≠.
E2EPT:       .BLK      1                     ;  LAST ENTRY OF
;                                            ≠ER2TB≠.
```

```
ERCAD:   .BLK    1          ; WHICH PASS IS
;                            REQUIRED.
ENDFG:   .BLK    1          ; WHETHER CURR.
;                            PASS HAS ENDED.
ER1TB:           ER1T1      ; ADD. OF PASS1
;                            ERROR TABLE.
ER2TB:           ER2T1      ; ADD. OF PASS2
;                            ERROR TABLE.
INPDV:   .BLK    1          ;INP. DV.
INSLN:           INSL1      ; ADD. OF INPUT
;                            LINE.
LBFLG:   .BLK    1          ; WHETHER LABEL
;                            IS PRESENT IN
;                            CURR. LINE.
LCFLG:   .BLK    1          ; WHETHER LOCTR
;                            HAS TO BE
;                            LISTED.
LNGTH:   .BLK    1          ; LENGTH OF CURR
;                            -ENT INST.
LNCTR:   .BLK    1          ; CURR.LINE NO.
LOCAL:   .BLK    1          ; NO. OF WDS. IN
;                            OBJECT BUFFER.
LOCTN:   .BLK    1          ; VALUE ENTERED
;                            IN THE LOC.
;                            FIELD.
LOCTR:   .BLK    1          ; ADD. OF CURR.
;                            INSTRUCTION.
LTFLG:   .BLK    1          ;WHETHER LISTING
;                            IS REQD.
MOPTB:           MOPT1      ; ADD. OF M.O.T.
NAME:            NAME1      ; ADD. OF CURR.
;                            NAME.
NEMOT:           110        ; NO. OF ENTRIES
;                            IN M.O.T.
NOSYM:   .BLK    1          ; NO. OF SYMBOLS
;                            IN SYM.TABLE.
NOWDS:           25         ; NO. OF WDS.
;                            PER DATA REC.
OPBEG:   .BLK    1          ;POINTS TO OPCODE
;                            IN INPUT LINE.
OPDTB:           OPDT1      ; ADD. OF DEF.
;                            OPTIONS TABLE.
OPFTB:           OPFT1      ; ADD. OF CURR.
;                            OPTIONS TABLE.
OPRBG:   .BLK    1          ; POINTS TO OPER
;                            -ERANDS IN
;                            INPUT LINE.
```

```
OUTDV:    .BLK    1                       ;OUTPT DV.
P1ERR:    .BLK    1                       ;ERROR WD CORR.
                                           TO PASS1.
PAGNO:    .BLK    1                       ; CURR. PAGE OF
                                           LISTING.
PNFLG:    .BLK    1                       ; WHETHER BUFFER
                                           HAS TO BE
                                           FLUSHED.
POEPT:            PCPT1+41                ; P.O.T. END
                                           POINTER.
POPTB:            PCPT1                   ; ADD. OF P.O.T.
PRERR:    .BLK    1                       ;ERROR WD OF
                                           CURR. PASS.
PRPTR:    .BLK    1                       ; POINTS TO NEXT
                                           CH. IN PRTLN.
PRTLN:            PRTL1                   ; ADD. OF PRINT
                                           LINE.
PSFLG:    .BLK    1                       ; WHETHER PASS
                                           1 OR 2.
SCFLG:    .BLK    1                       ; WHETHER LIST-
                                           -ING IS NECC.
                                           IN THIS SCAN.
SYMTB:            SYMT1                   ;ADD. OF SYMTB.
                                           (SYMBOL TABLE).
VALUE:    .BLK    1                       ; VALUE TO BE
                                           PRINTED IN A
                                           RMB/EQU INST.
VLFLG:    .BLK    1                       ; WHETHER VALUE
                                           HAS TO BE
                                           PRINTED.
WSPCE:            WSPC1                   ; ADD. OF SPACE
                                           WHERE INST. IS
                                           ASSEMBLED.

;*************************************************************

          .LOC    200

;*************************************************************

; THE FOLLOWING REPRESENT THE ADDRESSES OF VARIOUS
; ROUTINES USED ,SO THAT THEY CAN BE ACCESSED
; INDIRECTLY THRO THE BASE PAGE.
```

```
:
:**************************************************
:
:
ADD88:          AD188
ASSMB:          ASSM
BDC46:          BC146
BHX47:          BH147
BIN15:          BI115
BNY93:          BN193
CAC22:          CA122
CEF14:          CE114
CHCL6:          CH106
CHK73:          CH173
CLB76:          CL176
CVT84:          CV184
CVT96:          CV196
DEC91:          DE191
DI800:          D1800
DIV90:          DI190
EBN36:          EB136
ELC29:          EL129
END52:          EN152
ENE26:          EN126
ENET9:          EN109
EPT83:          EP183
EQU53:          EQ153
ERL28:          ER128
ESC32:          ES132
EVL97:          EV197
EWS31:          EW131
EX802:          E1802
FCB54:          FC154
FCC55:          FC155
FDB56:          FD156
FVF34:          FV134
GCH81:          GC181
GET78:          GE178
GLB41:          GL141
GNC87:          GN187
HEX94:          HE194
HSH80:          HS180
IBN35:          IB135
IMM99:          IM199
IN801:          I1801
INITO:          IN100
INFS4:          IN104
```

```
INT13:          IN113
LAC21:          LA121
MPY72:          MP172
LOC30:          LC130
LSM18:          LS118
MOD85:          MC185
NAM57:          NA157
NEF17:          NE117
OCT92:          OC192
OPT58:          OP158
ORG59:          OR159
ORF77:          OR177
PAG60:          PA160
PASS1:          PA101
PASS2:          PA102
PCH37:          PC137
PEO20:          PE120
PLF08:          PL108
PN803:          P1803
PNL12:          PN112
PROP7:          PR107
PFT82:          PR182
PSC10:          PS110
PSY39:          PS139
PUN40:          PU140
PUT75:          PU175
RED74:          RE174
RMB61:          RM161
RNCD5:          RN105
SAB98:          SA198
SCP23:          SC123
SMT44:          SM144
SPC62:          SP162
SPM25:          SP125
SPT43:          SP143
SST51:          SS151
STAC4:          S1804
STB79:          ST179
SUB89:          SU189
SUM50:          SU150
SYM95:          SY195
UPD11:          UP111
VLB42:          VL142
VTR24:          VT124
WDUP3:          WD103
WRT86:          WR186
;
```

```
;*****************************************************
;
; THE FOLLOWING REPRESENT THE ADDRESSES OF THE SERVICE
; ROUTINES FOR THE NINE DIFFERENT TYPES OF MACHINE-
; -OPERATION-CODES OF THE M6800.
;
;*****************************************************

ADC63:          AD163
STA64:          ST164
ASL65:          AS165
PSH66:          PS166
CPX67:          CP167
STS68:          ST168
JMP69:          JM169
BCC70:          BC170
ABA71:          AB171

;*****************************************************

        .LOC    2000

;*****************************************************

; NAME.                         ASSMB
; FUNCTION.                     CROSS ASSEMBLER.
; INPUT.                        SOURCE-CODE AS DEFINED IN
;                               THE REPORT.
; OUTPUT.                       SOURCE-LISTING.
;                               OBJECT CODE IN THE MOTOROLA
;                               PAPER-TAPE FORMAT.
; CALLS.                        MOD65,PASS1,PASS2
; CALLED-BY.                    NONE.
; GLOBAL-VARIABLES-USED.        ENDAD.
; GLOBAL-VARIABLES-CHANGED.
```

```
;
;  ERROR-BITS-SET.                    NONE.
;
;                                     NONE.
;
;
;  MOD85 IS CALLED TO DECIDE  PASS1 OR PASS2 IS TO BE
;  CALLED.MOD85 ALSO SETS FLAGS TO DENOTE WHETHER OBJECT
;  CODE OR LISTING IS NECC.
;
ASSM:     JSRS     MOD85
          LDA      0,ENDAD
          MOV      0,0,SZR
;
;  ENDAD=1,PASS2 IS NECC
;
          JMP      NEXT
;
;  ENDAD=0,PASS1
;
          JSRS     PASS1
          JMP      ASSM
;
;  PASS2.
;
NEXT:     JSRS     PASS2
          JMP      ASSM
;
;*****************************************************
;
;  NAME.                              INIT0.
;  FUNCTION.                          INITIALISES VARIABLES BEFOR
;                                     PASS1.
;  INPUT.                             NONE.
;  OUTPUT.                            NONE.
;  CALLS.                             NONE.
;  CALLED-BY.                         PASS1.
;  GLOBAL-VARIABLES-USED.             OIRAD,DEFNM,ER1TB,NAME,
;                                     OPDTB,OPFTB,SYMTB.
```

```
; GLOBAL-VARIABLES-CHANGED.
;                                BNFLG,DEFN1,DIRA1,DREPT,
;                                E1EPT,ENDFG,ER1T1,LIOCTR,
;                                LOCTN,LOCTR,LTFLG,NAME1,
;                                NOSYM,OPFT1,SYMT1.
; ERROR-BITS-SET.
;                                NONE.
;
; SPACE FOR SAVING ACC. CONTENTS.
;
AC000:    .BLK    1
AC100:    .BLK    1
AC200:    .BLK    1
AC300:    .BLK    1
;
; CONSTANTS,COUNTERS.
;
CN000:            0
CN100:            1
CN200:            62                    ; DEC. 50.
CN300:    .BLK    1
CN400:            6
CN500:    .BLK    1
CN600:            3
CN700:            5
CN800:            400                   ;DEC  256
CN900:            4                     ;DEC  4
CTR00:    .BLK    1                     ;COUNTER
;
; SAVE ACCS.,ENTRY.
;
IN100:    STA     0,AC000
          STA     1,AC100
          STA     2,AC200
          STA     3,AC300
;
; INITIALISE BNFLG,LOCTR,LTFLG TO 0.
; INITIALISE ENDFG,LOCTN, TO 0.
;
          LDA     0,CN000
          STA     0,BNFLG
          STA     0,ENDFG
          STA     0,LOCTN
          STA     0,LOCTR
          STA     0,LTFLG
```

```
;
; INITIALISE DIRAD,ERITE TO ALL ZEROES.
;
        LDA     3,CM200
        STA     3,CM300
        LDA     2,DIRAD
        LDA     3,ER1TB
;
        STA     0,0,2
        STA     0,0,3
        INC     2,2
        INC     3,3
        DSZ     CM300
        JMP     .-5
;
;
; INITIALISE LNCTR,PSFLG,TO 1.
;
        LDA     0,CM100
        STA     0,LNCTR
        STA     0,PSFLG
;
; INITIALISE DREPT
;
        LDA     0,DIRAD
        STA     0,DREPT
;
; INITIALISE E1EPT.
;
        LDA     0,ER1TB
        STA     0,E1EPT
;
; INITIALISE NAME TO DEFNM.
;
        LDA     2,DEFNM
        LDA     3,NAME
        LDA     0,0,2
        STA     0,0,3
        LDA     0,1,2
        STA     0,1,3
        LDA     0,2,2
        STA     0,2,3
;
;
; INITIALISE OPTION FLAG TABLE (OPFTB) TO
; DEFAULT VALUES (OPDTB).
```

```
;
; CN400 CONTAINS DEC. 6.
;
        LDA     0,CN400
        STA     0,CN500
        LDA     2,OPDTB
        LDA     3,OPFTB
;
        LDA     0,0,2
        STA     0,0,3
        INC     2,2
        INC     3,3
        DSZ     CN500
        JMP     .-5
;
; INITIALISE NOSYM.
; CN600 HAS VALUE DECIMAL 3;
;
        LDA     2,OPDTB
        LDA     0,CN600
        ADD     0,2
        LDA     0,0,2
        STA     0,NOSYM
;
; INITIALISE SYMBOL TABLE,256 ENTRIES,FIRST WORD OF
; EACH ENTRY IS CLEARED.
;
        LDA     0,CN300                 ;DEC 256
        STA     0,CTR00
;
;
        LDA     3,SYMTB
        LDA     1,CN900
        LDA     0,CN000                 ; VALUE 0
;
        STA     0,0,3                   ;CLR NEXT ENTRY.
        ADD     1,3                     ;ADD 4
        DSZ     CTR00
        JMP     .-3
;
; RESTORE ACC CONTENTS AND RETURN.
;
        LDA     0,AC000
        LDA     1,AC100
        LDA     2,AC200
        JMPS    AC300
;
```

```
;*****************************************************
;NAME.                          PASS1.
;FUNCTION.                      PERFORMS THE PASS1 FUNCTION
;                               OF SETTING UP SYMBOL TABLE
;                               AND ERROR TABLE.ALSO SETS
;                               UP ≠DIRECT ADDRESS≠TABLE.
;INPUT.                         SOURCE CODE.
;OUTPUT.                        SYMBOL TABLE,ERROR TABLE
;                               AND ≠DIRECT ADDRESS≠ TABLE.
;CALLS.                         INIT0,INPS4,RNC05,CHCL6,
;                               PROP7,ENET3,PSC10,UPD11,
;                               NER17.
;CALLED-BY.                     ASSMB.
;GLOBAL-VARIABLES-USED.         CMFLG,ENDFG.
;GLOBAL-VARIABLES-CHANGED.      NONE.
;ERROR-BITS-SET.                NONE.
;
;LOCATIONS FOR STORING ACC CONTENTS.
;
AC001:  .BLK    1
AC101:  .BLK    1
AC201:  .BLK    1
AC301:  .BLK    1
;
;STORE ACC CONTENTS.
;
PA101:  STA     0,AC001
        STA     1,AC101
        STA     2,AC201
        STA     3,AC301
;
;INITIALISE MISC. VARIABLES.
;
        JSRS    INIT0
;
```

```
;
; CALL NECC. ROUTINES UNTIL END PSEUDO-OP
; IS ENCOUNTERED.
;
LB101:  JSR$    INPS4
        JSR$    ENCD5
        JSR$    CHCL6
;
; CHECK WHETHER COMMENT OR NOT
;
        LDA     0,CMFLG
        MOV     0,0,SNR
;
; NOT A COMMENT.
;
        JSR$    PROP7
        JSR$    ENET9
        JSR$    PSC10
        JSP$    UPD11
;
; CHECK FOR END OF PASS.
;
        LDA     0,ENDFG
        MOV     0,0,SNR
        JMP     LB101
;
; END OF PASS-1.
;
; LIST OUT TOTAL NO. OF ERRORS.
;
        JSR$    NER17
;
; RESTORE ACCS. AND RETURN.
;
        LDA     0,AC001
        LDA     1,AC101
        LDA     2,AC201
        JMP$    AC301
;
;*****************************************************
;
; NAME.                         INI13
; FUNCTION.                     INITIALISES THE NECC. VARIA
```

```
;
;  INPUT.                          -BLES BEFORE PASS2.
;
;  OUTPUT.                         NONE.
;
;  CALLS.                          NONE.
;
;  CALLED-BY.                      PASS4.
;
;  GLOBAL-VARIABLES-USED.          PASS2.
;
;  GLOBAL-VARIABLES-CHANGED.       BNOUT,ER2TB.
;                                  BNOL1;E2EPT,ER2TB,LNCTR,
;                                  LOCAL,LOCTN,LOCTR,PAGNO,
;                                  PSFLG,ENDFG.
;
;  ERROR-BITS-SET.                 NONE.
;
;
;  SPACE FOR SAVING ACCS.
;
AC013:    .BLK    1
AC113:    .BLK    1
AC213:    .BLK    1
AC313:    .BLK    1
;
;  CONSTANTS.
;
CNC13:            50                    ; DEC. 40.
CN113:            62                    ; DEC. 50
CTR13:    .BLK    1
;
;  SAVE ACCS. ENTRY.
;
IN113:    STA     0,AC013
          STA     1,AC113
          STA     2,AC213
          STA     3,AC313
;
;  INITIALISE E2EPT.
;
          LDA     0,ER2TB
          STA     0,E2EPT
;
;  INITIALISE LOCTR,LOCAL,LOCTN,PAGNO,ENDFG TO 0.
;
          SUB     0,0
```

```
        STA     0,LOCAL
        STA     0,LOCTR
        STA     0,LOCTN
        STA     0,PAGNO
        STA     0,ENDFG
;
;;; INITIALISE LNCTR TO 1.
;
        INC     0,0
        STA     0,LNCTR
;
;;; INITIALISE PSFLG TO 2.
;
        INC     0,0
        STA     0,PSFLG
;
;;; INITIALISE ER2TB,CN113=5.
;
        LDA     0,CN113
        STA     0,CTR13
;
        LDA     2,ER2TB
        SUB     0,0
;
LB113:  STA     0,0,2
        INC     2,2
        DSZ     CTR13
        JMP     LB113
;
;;; INITIALISE BNOUT.CN213=4.
;
        LDA     0,CN213
        STA     0,CTR13
;
        LDA     2,BNOUT
        SUB     0,0
;
LB213:  STA     0,0,2
        INC     2,2
        DSZ     CTR13
        JMP     LB213
;
;;; CALL ON ROUTINE TO PUNCH OUT START RECORD.
;
        JSRS    PN303
;;;
```

```
; CALL ON ROUTINE TO PRINT OUT HEADING.
;
        JSRS    PAGEJ
;
; RESTORE ACCS AND RETURN.
;
        LDA     0,AC013
        LDA     1,AC113
        LDA     2,AC213
        JMPS    AC313
;
;*****************************************************
;
; NAME.                                 PASS2
; FUNCTION.                             PRODUCES SOURCE-LISTING,
;                                       OBJECT CODE,SYMBOL TABLE.
; INPUT.                                SYMBOL TABLE,SOURCE-CODE,
;                                       #DIRECT-ADDRESS TABLE.
; OUTPUT.                               OBJECT CODE,SOURCE LISTING,
;                                       ERROR LISTING,SYMBOL TABLE.
; CALLS.                                INT13,INPS4,RNC05,CHCL6,
;                                       PROP7,CER14,ENET9,BIN15,
;                                       PSC10,UPD11,WDUP3.
; CALLED-BY.                            ASSMB.
; GLOBAL-VARIABLES-USED.                CMFLG,ERRFG.
; GLOBAL-VARIABLES-CHANGED.             NONE.
; ERROR-BITS-SET.                       NONE.
;
; LOCATIONS FOR STORING ACC CONTENTS.
;
AC002:  .BLK    1
AC102:  .BLK    1
AC202:  .BLK    1
AC302:  .BLK    1
;
```

```
;
; SAVE ACC CONTENTS.
;
PA102:  STA     0,AC002
        STA     1,AC102
        STA     2,AC202
        STA     3,AC302
;
; INITIALISE MISC. VARIABLES.
;
        JSR$    INT13
;
; SCAN SOURCE CODE LINE BY LINE UNTIL
; END PSEUDO-OP IS ENCOUNTERED.
;
LEL02:  JSR$    INPS4                   ; INITIALISE.
;
; READ FROM INPUT
;
        JSR$    RNCD5
;
; CHECK COL 1 AND TAKE ACTION.
;
        JSR$    CHCL6
;
; CHECK WHETHER COMMENT OR NOT.
;
        LDA     0,CMFLG
        MOV     0,0,SNR
;
; NOT A COMMENT,PROCESS OP-CODE,CHECK ERROR WORD,
; ENTER INTO ERROR TABLE.
;
        JSR$    PROP7
        JSR$    CER14
        JSR$    ENET9
;
; CALL ON ROUTINES TO PRODUCE OBJECT CODE/LISTING.
;
        JSR$    BIM15
        JSR$    PSC10
;
; UPDATE COUNTERS.
;
        JSR$    UPD11
;
```

```
;
; CHECKK FOR END OF PASS
;
        LDA     0,ENDFG
        IOV     0,0,SNR
        JMP     LBL02
;
; END OF PASS 2,CLEAN-UP,RESTORE ACCS AND RETURN.
;
        JSRS    WDUP3
;
        LDA     0,AC002
        LDA     1,AC102
        LDA     2,AC202
        JMPS    AC302
;
;*****************************************************
;
; NAME.                         WDUP3
; FUNCTION.                     DOES THE CLEANING UP.
; INPUT.                        SYMBOL TABLE,ERROR TABLE.
; OUTPUT.                       EOF RECORD,SYMBOL TABLE,
;                               NO. OF ERRORS.
; CALLS.                        NEP17,LSM16,PED20.
; CALLED-BY.                    PASS2.
; GLOBAL-VARIABLES-USED.        NONE.
; GLOBAL-VARIABLES-CHANGED.     NONE.
; ERROR-BITS-SET.               NONE.
;
;  LOCATIONS TO STORE ACC CONTENTS.
AC303:  .BLK    1
; SAVE ACC CONTENTS
WD103:  STA     3,AC303
;
```

```
PUNCH OUT EOF RECORD.
        JSR$    PE020
PRINT OUT TOTAL NO. OF ERRORS.
        JSR$    NER17
LIST SYMBOL TABLE.
        JSR$    LSM16
RESTORE ACCS AND RETURN.
        JMP$    AC333
```

****

NAME. INPS4

FUNCTION. INITIALISE VARIABLES ON EACH SCAN.

INPUT. NONE.

OUTPUT. NONE.

CALLS. NONE.

CALLED-BY. PASS1,PASS2.

GLOBAL-VARIABLES-CHANGED. ABFLG,ABVAL,CHPTR,CMBEG, CMFLG,INSL1,LBFLG,LCFLG, LENTH,OPBEG,OPFLG,P1ERR, PNFLG,PREPB,PRPTR,PRTL1, SCFLG,VALUE,VLFLG,WSPC1.

GLOBAL-VARIABLES-USED. WSPCE,PRTLN,INSLN.

ERROR-BITS-SET. NONE.

SPACE FOR SAVING ACC CONTENTS.

```
AC004:    .BLK    1
AC104:    .BLK    1
AC204:    .BLK    1
AC304:    .BLK    1
;
; CONSTANTS,COUNTERS.
;
CN204:            2
CN304:            50                          ;DEC 40
CN404:            74                          ;DEC 60
CN504:            20040                       ; 2 BLANKS
CTR04:    .BLK    1                           ;COUNTER
;
; SAVE ACC CONTENTS.
;
IN104:    STA     0,AC004
          STA     1,AC104
          STA     2,AC204
          STA     3,AC304
;
;
; INITIALISE CMFLG,LBFLG,LENTH,PNFLG,PRERR,ABFLG,
; BSVAL,VLFLG,VALUE ,VLFLG TO 0.
;
          SUB     0,0
          STA     0,CMFLG
          STA     0,LBFLG
          STA     0,LENTH
          STA     0,PNFLG
          STA     0,PRERR
          STA     0,ABFLG
          STA     0,BSVAL
          STA     0,VALUE
          STA     0,VLFLG
;
; INITIALISE CHPTR,CMBEG,PRPTR,OPBEG,OPRBG,
; LCFLG,SCFLG TO 1.
;
          INC     0,0
          STA     0,CHPTR
          STA     0,CMBEG
          STA     0,PRPTR
          STA     0,OPBEG
          STA     0,OPRBG
          STA     0,SCFLG
          STA     0,LCFLG
;
```

```
;
; INITIALISE WSPCE TO 0.
;
        LDA     2,WSPCE
        STA     0,0,2
        STA     0,1,2
;
; INITIALISE PRTLN TO ALL BLANKS.CN404=60,CN504=BLANKS.
;
        LDA     1,CN404
        STA     1,CTR04
;
        LDA     2,PRTLN
        LDA     0,CN504
;
LB104:  STA     0,0,2
        INC     2,2
        DSZ     CTR04
        JMP     LB104
;
; INITIALISE INSLN TO ALL BLANKS.CN304=40,CN504=BLANKS.
;
        LDA     0,CN304                 ;DEC 40.
        STA     0,CTR04
;
        LDA     0,CN504                 ;BLANKS
        LDA     2,INSLN
;
LB204:  STA     0,0,2
        INC     2,2
        DSZ     CTR04
        JMP     LB204
;
;
; SET VALUE FOR PIERR IF PASS2.CN204=2.
;
        LDA     0,PSFLG
        LDA     1,CN204
;
; CHECK FOR PASS=2
;
        SUB     0,1,SZR
        JMP     END04                   ; PASS IS 1,
;                                         DO NOTHING.
; PASS IS 2.SET VALUE FOR PIERR.
;
        LDA     1,LNCTR
```

```
        LDA     2,ER1TE
        LDA     3,E1EPT
;
LB304:  SUBE    2,3,SNR                 ; CHECK FOR
;                                       ; END OF TABLE.
        JMP     NFD04                   ; NOT FOUND.
;
; WHOLE TABLE HAS NOT BEEN SEARCHED.
;
        LDA     0,0,2
        SUBE    0,1,SNR                 ; EQUALITY TEST.
        JMP     END04                   ; FOUND.
        INC     2,2
        INC     2,2                     ; CHECK NEXT
;                                       ; ENTRY.
        JMP     LB304
;
; FOUND IN TABLE. INITIALISE P1ERR.
;
END04:  INC     2,2
        LDA     0,0,2
        STA     0,P1ERR
        JMP     END04
;
; NOT FOUND,SET TO 0.
;
NFD04:  SUB     0,0
        STA     0,P1ERR
;
; RESTORE ACC CONTENTS AND RETURN.
;
END04:  LDA     0,AC004
        LDA     1,AC104
        LDA     2,AC204
        JMPS    AC304
;
;*****************************************************
;
; NAME.                         RNCC5.
;
; FUNCTION.                     GET NEXT CARD FROM INPUT
;                               DEVICE INTO INSLN.
;
; INPUT.                        NONE.
;
; OUTPUT.
```

```
;
; CALLS.                          INSLN.
; CALLED-BY.                      REQ74,CHR73.
; GLOBAL-VARIABLES-USED.          PASS1,PASS2.
; GLOBAL-VARIABLES-CHANGED.       SCIND,ECODV,INSLN.
; ERROR-BITS-SET.                 INPDV,OUTDV,INSL1.
;                                 NONE.
;
; LOCATIONS FOR SAVING ACC CONTENTS.
;
AC005:    .BLK    1
AC105:    .BLK    1
AC205:    .BLK    1
AC305:    .BLK    1
;
; CONSTANTS. 40,BLANK
;
CN005:            51                        ; DEC  41
BLK05:            40                        ;  BLANK
CTR05:    .BLK    1
;
; SAVE ACC CONTENTS.
;
PN105:    STA     0,AC005
          STA     1,AC105
          STA     2,AC205
          STA     3,AC305
;
; SET INPUT AND OUTPUT DEVICE CODES.
;
          LDA     0,SCIND
          STA     0,INPDV
          LDA     0,ECODV
          STA     0,OUTDV
;
; SET COUNTER VALUE.
; CN005= 40 DEC
;
          LDA     0,CN005
          STA     0,CTR05
;
          LDA     2,INSLN
```

```
;
LB10S:  SUB     0,0                     ;CLEAR ACC 0.
        JSR@    RED74
;
; ACC 0 HAS CH IN RIGHT HALF.
;
        JSR@    CHK73                   ;CHECK IF CR,LF
        JMP     CR10S                   ;CH IS CR,LF
        MOVS    0,1                     ; CH. IS OKAY,
;                                         SWAP AND MOVE.
        SUB     0,0                     ; CLEAR ACC0
        JSR@    RED74                   ; GET NEXT CH.
        JSR@    CHK73
        JMP     CR20S                   ;CR OR LF
        ADD     0,1                     ;CH IS OKAY.
        STA     1,0,2                   ;STORE IN INSLN.
        INC     2,2
        DSZ     CTR0S
        JMP     LB10S
;
; COUNT OF 80 CHARACTERS.
;
        JMP     END0S
;
; ODD CHARACTER IS CR.SO NO CH. TO BE SAVED.
;
CR10S:  JMP     END0S
;
; EVEN CHARACTER IS CR.SAVE 1 CH AND RETURN.
;
CR20S:  MOV     1,0
        LDA     1,BLK0S
        ADD     1,0
        STA     0,0,2
;
;
; RESTORE ACC AND RETURN.
;
END0S:  LDA     0,AC00S
        LDA     1,AC10S
        LDA     2,AC20S
        JMP@    AC30S
;
;*****************************************************
;
; NAME.
```

```
;
;  FUNCTION.                    CHK73
;                               CHECKS WHETHER CH. IN
;                               ACC 0 IS CR,LF OR FF.
;  INPUT.
;                               CH. RIGHT ADJUSTED IN
;                               ACC 0 WITH BIT 8 SET TO 0 .
;  OUTPUT.
;                               IF CH. IS CR,LF OR FF,
;                               PUT OUT A LF AND TAKE
;                                 ≠AC 3 ≠RETURN.
;                               OTHERWISE TAKE
;                               ≠ACC 3 + 1≠ RETURN.
;  CALLS.
;                               PUT75
;  CALLED-BY.
;                               RNCC5.
;  GLOBAL-VARIABLES-USED.
;                               NONE.
;  GLOBAL-VARIABLES-CHANGED.
;                               NONE.
;  ERROR-BITS-SET.
;                               NONE.
;  LOCATIONS FOR SAVING ACC CONTENTS.
AC073:    .BLK    1
AC173:    .BLK    1
AC273:    .BLK    1
AC373:    .BLK    1
;
;  CONSTANTS.
;
CN173:              015                     ;CR
CN273:              014                     ;FF
CN373:              012                     ;LF
;
;  SAVE ACC CONTENTS.
;
CH173:    STA       0,AC073
          STA       1,AC173
          STA       2,AC273
          STA       3,AC373
;
;
;  CHECK TO SEE WHETHER CH. IS FF,CR OR LF.
;
```

```
        LDA     1,CN173                 ; CR
        SUBE    0,1,SNR
        JMP     CRT73                   ; CH. IS CR.
;
        LDA     1,CN273                 ; FF
        SUBE    0,1,SNR
        JMP     CRT73                   ;CH IS FF.
        LDA     1,CN373                 ; LF
        SUBE    0,1,SNR
        JMP     CRT73                   ; CH. IS LF.
;
; CH IS NEITHER. TAKE NORMAL RETURN
;
        INC     3,3
        STA     3,AC373
        JMP     END73
;
; CH IS FF OR CR.PUT OUT A LF AND TAKE AC3 RETURN.
;
CRT73:  LDA     0,CN373                 ; LF
        JSRS    PUT75
;
END73:  LDA     0,AC073
        LDA     1,AC173
        LDA     2,AC273
        JMPS    AC373
;
;*****************************************************
;
; NAME.                         CHCL6.
;
; FUNCTION.                     CHECKS COL 1 AND DECIDES
;                               IF LINE IS A COMMENT OR
;                               LABEL AND TAKE CORR. ACTION
;
; INPUT.                        INSLN.
;
; OUTPUT.                       NONE.
;
; CALLS.                        CAC21,CAC22,GCH31.
;
; CALLED-BY.                    PASS1,PASS2.
;
; GLOBAL-VARIABLES-USED.        NONE.
;
; GLOBAL-VARIABLES-CHANGED.
```

```
;
; ERROR-BITS-SET.                    NONE.
;                                    NONE.
;
;
; LOCATIONS FOR SAVING ACC CONTENTS.
;
AC006:    .BLK    1
AC106:    .BLK    1
AC206:    .BLK    1
AC306:    .BLK    1
;
; CH CODES FOR BLANK,* ,AND MASK
;
CH106:              40                  ; BLANK
CH206:              52                  ; *
CH306:              377                 ; MASK
;
; SAVE ACC CONTENTS.
;
CH106:    STA       0,AC006
          STA       1,AC106
          STA       2,AC206
          STA       3,AC306
;
; GET FIRST CHARACTER.
;
          JSRS      GCH61
;
; ACC CONTAINS FIRST CHARACTER.CHECK THE CHARACTER.
;
          LDA       1,CH106             ; BLANK
          SUBE      0,1,SNR
;
; CHARACTER IS A BLANK. TAKE NO ACTION.
;
          JMP       END06
;
; CHECK WHETHER * OR NOT.
;
          LDA       1,CH206             ; *
          SUBE      0,1,SNR
          JMP       CMT06               ; COMMENT.
;
          JSRS      LAC21               ; LABEL.
          JMP       END06
;
```

```
; COMMENT ACTION.
CMTC6:  JSRS    CAC22

; RESTORE ACC AND RETURN.
ENDC6:  LDA     0,AC006
        LDA     1,AC106
        LDA     2,AC206
        JMPS    AC306

;*****************************************************

; NAME.                          CAC22.
; FUNCTION.                      SET CMFLG.
; INPUT.                         NONE.
; OUTPUT.                        NONE.
; CALLS.                         NONE.
; CALLED-BY.                     CHCL6.
; GLOBAL-VARIABLES-USED.         NONE.
; GLOBAL-VARIABLES-CHANGED.      CMFLG,LCFLG.
; ERROR-BITS-SET.                NONE.

; LOCATIONS FOR STORING ACC CONTENTS.
AC022:  .BLK    1
CNC22:          1

; SAVE ACCS AND SET FLAG.
CA122:  STA     0,AC022

        LDA     0,CN022
        STA     0,CMFLG
;
```

```
; SET LCFLG TO 0.
        SUB     0,0
        STA     0,LCFLG
;
; RESTORE ACC CONTENTS.
        LDA     0,AC022
        JMP     0,3
;
;*****************************************************
;
; NAME.                             LAC21.
; FUNCTION.                         SET LBFLG,CHECK LABEL,
;                                   AND TAKE ACTION.
; INPUT.                            INSLN.
; OUTPUT.                           NONE.
; CALLS.                            GL341,VLS42.
; CALLED-BY.                        CHCL6.
; GLOBAL-VARIABLES-USED.            NONE.
; GLOBAL-VARIABLES-CHANGED.         LBFLG.
; ERROR-BITS-SET.                   NONE.
;
; LOCATIONS FOR SAVING ACC CONTENTS.
AC021:  .BLK    1
AC121:  .BLK    1
AC221:  .BLK    1
AC321:  .BLK    1
; CONSTANTS.
CN021:          1
; SAVE ACC CONTENTS.
```

```
;
LA121:  STA     0,AC021
        STA     1,AC121
        STA     2,AC221
        STA     3,AC321
;
;SET LABEL FLAG.
;
        LDA     0,CN021                 ; DEC 1
        STA     0,LBFLG
;
;GET LABEL. IE. SET CHPTR TO POINT TO END
;OF LABEL(CH AFTER LABEL).
;
        JSR@    GL341
;
;CHPTR HAS BEEN SET.CALL ON ROUTINE TO CHECK
;VALIDITY OF LABEL,AND IF VALID TAKE ACTION.
;
        JSR@    VL342
;
;RESTORE ACC CONTENTS AND RETURN.
;
        LDA     0,AC021
        LDA     1,AC121
        LDA     2,AC221
        JMP@    AC321
;
;*****************************************************
;
;NAME.                          CL376.
;FUNCTION.                      CHECKS WHETHER CH IN
;                               ACC 0 IS ALPHANUMERIC.
;INPUT.                         CH. IN RIGHT HALF OF ACC 0.
;OUTPUT.                        #AC 3 + 1 #RETURN IF ALPHA
;                               #AC 3 # RETURN OTHERWISE.
;CALLS.                         NONE.
;CALLED-BY.                     VL842.
;GLOBAL-VARIABLES-USED.         NONE.
```

```
; GLOBAL-VARIABLES-CHANGED.                    NONE.
; ERROR-BITS-SET.                              NONE.
;
; LOCATIONS FOR STORING ACC CONTENTS.
;
AC076:    .BLK    1
AC176:    .BLK    1
AC276:    .BLK    1
AC376:    .BLK    1
;
; CONSTANTS.
;
CN076:            60                           ;≠0≠
CN176:            71                           ;≠9≠
CN276:            101                          ;≠A≠
CN376:            132                          ;≠Z≠
;
; SAVE ACC CONTENTS.
;
CL176:    STA     0,AC076
          STA     1,AC176
          STA     2,AC276
          STA     3,AC376
;
; CHARACTER IS IN RIGHT HALF OF ACC.
;
          LDA     1,CN076                      ; ≠0≠
          SUBLE   1,0,SZC
          JMP     NXT76                        ; NOT NUMERIC.
;
          LDA     1,CN176                      ;≠9≠
          SUBLE   0,1,SZC                      ; CHECK.
          JMP     NXT76                        ; NOT NUMERIC.
          JMP     VLD76                        ; VALID CH,
;                                              NUMERIC.
; CHECK WHETHER ALPHABETIC.
;
NXT76:    LDA     1,CN276                      ; ≠A≠
          SUBLE   1,0,SZC                      ; CHECK
          JMP     ERR76                        ; NOT ALPHA-
;                                              NUMERIC,ERROR.
          LDA     1,CN376                      ; ≠Z≠
```

```
          SUBLE   0,1,SZC
          JMP     ERR76                   ; NOT ALPHA-
;                                         NUMERIC,ERROR.
          JMP     VLD76                   ; ALPHABETIC.
;
; VALID CHARACTER,TAKE AC3+1 RETURN.
;
VLD76:    LDA     0,AC376
          INC     0,0
          STA     0,AC376
;
ERR76:    LDA     0,AC076
          LDA     1,AC176
          LDA     2,AC276
          JMPS    AC376
;
;*****************************************************
;
; NAME.                          VLB42.
; FUNCTION.                      CHECKS VALIDITY OF LABEL
;                                AND TAKES CORR. ACTION.
; INPUT.                         CHRTR POINTS TO CH. AFTER
;                                LABEL.
; OUTPUT.                        NONE.
; CALLS.                         SST51,ERR26,CLB76.
; CALLED-BY.                     LAC21.
; GLOBAL-VARIABLES-USED.         INSLN.
; GLOBAL-VARIABLES-CHANGED.      NONE.
; ERROR-BITS-SET.                BIT 0 (I).
;
; LOCATIONS FOR SAVING ACC CONTENTS.
AC042:    .BLK    1
AC142:    .BLK    1
AC242:    .BLK    1
```

```
AC342:   .BLK    1
;
; CONSTANTS AND COUNTERS.
;
CN042:           10                     ;DEC 8
CN242:           377                    ;MASK
CN342:           101                    ;  'A'
CN442:           132                    ;'Z'
CTR42:   .BLK    1
;
; SAVE ACC CONTENTS.
;
VL142:   STA     0,ACC42
         STA     1,AC142
         STA     2,AC242
         STA     3,AC342
;
; CHECK FOR VALIDITY OF LENGTH.
;
         LDA     0,CHPTR
         LDA     1,CN042                ;DEC 8
         SUBLE   1,0,SNC
         JMP     ERR42                  ;INVALID LENGTH
;
; CHECK FOR FIRST CHARACTER ALPHABETIC.
;
         LDA     2,INSLN
         LDA     0,0,2
         MOVS    0,0
         LDA     1,CN242                ; MASK
         AND     1,0
;
         LDA     1,CN342                ;'A'
         SUBLE   1,0,SZC
         JMP     ERR42                  ; NOT ALPHA.
         LDA     1,CN442                ; 'Z'
         SUBLE   0,1,SZC
         JMP     ERR42                  ; NOT ALPHA.
;
; FIRST CH. IS OKAY,CHECK OTHERS FOR ALPHANUMERIC.
;
         LDA     0,CHPTR
         STA     0,CTR42
         DSZ     CTR42                  ; SET UP CTR.
;
;
         LDA     2,INSLN
```

```
;
LB142:  LDA     0,J,2
        MOVC    0,J
        LDA     1,CH242                 ; MASK
        AND     1,J
;
        JSR@    CLB76                   ; CHECK FOR
;                                       ALPHA-NUMERIC.
        JMP     ERR42                   ; NOT ALPHA.
;
        DSZ     CTR42
        JMP     .+2
        JMP     NXT42                   ; ALL CH CHECKED
;
        LDA     0,J ,2
        LDA     1,CH242                 ; MASK
        AND     1,J
;
        JSR@    CLB76                   ; CHECK
        JMP     ERR42                   ; NOT ALPHA.
        INC     2,2
        DSZ     CTR42
        JMP     LB142
;
; LABEL CHECKED AND VALID.
;
NXT42:  JSR@    SST51
        JMP     END42
;
; ERROR IN LABEL, ENTER IT.
;
ERR42:  SUB     0,J
        STA     0,ARG42
        JSR@    ERE26
ARG42:  .BLK    1
;
; RESTORE ACC AND RETURN.
;
END42:  LDA     0,AC042
        LDA     1,AC142
        LDA     2,AC242
        JMP@    AC342
;*****************************************************
;
;
; NAME.
```

```
;
;  FUNCTION.                        ENE26.
;
;  INPUT.                           SETS ERROR BIT.
;
;  OUTPUT.                          IN LOCATION AAC 3 + 1A
;
;  CALLS.                           NONE.
;
;  CALLED-BY.                       0?877.
;                                   SST51,VL042,FCC05,ST379,
;                                   SPM25,SYN90,EVL97,CIV90,
;                                   VTR24,BCC70,PSH56,ST364,
;                                   S4B96,RMB61,ORG59,END52,
;                                   EQU53.
;
;  GLOBAL-VARIABLES-USED.           NONE.
;
;  GLOBAL-VARIABLES-CHANGED.        PRERR.
;
;  ERROR-BITS-SET.                  BIT CORR. TO VALUE
;                                   IN LOC. AAC 3 + 1A.
;
;  LOCATIONS FOR SAVING ACC CONTENTS.
;
AC026:    .BLK    1
AC126:    .BLK    1
AC226:    .BLK    1
AC326:    .BLK    1
;
;  CONSTANTS FOR SETTING BIT IN PRERR.
;
EPS26:              1
                    2
                    4
                   10
                   20
                   40
                  100
                  200
                  400
                 1000
                 2000
                 4000
                10000
                20000
```

```
                        40000
                        100000
ERA26:                  ERS26
;
; SAVE ACC CONTENTS.
;
EN126:    STA       0,AC026
          STA       1,AC126
          STA       2,AC226
          STA       3,AC326
;
; LOAD FLAG NUMBER.
;
          LDA       0,0,3
;
; LOAD CORR. VALUE.
;
          LDA       2,ERA26
          ADD       0,2
          LDA       0,0,2
;
; FORM THE BIT IN.
;
          STA       0,AG126
          LDA       0,PRERR
          STA       0,AG226
          JSR@      ORR77
AG126:    .BLK      1
AG226:    .BLK      1
AG326:    .BLK      1
;
; SAVE IT BACK IN PRERR.
;
          LDA       0,AG326
          STA       0,PRERR
;
; RESTORE ACC AND RETURN.
;
          LDA       0,AC026
          LDA       1,AC126
          LDA       2,AC226
          LDA       3,AC326
;
          JMP       1,3
;
;*****************************************************
```

```
;
; NAME.                          REC74.
; FUNCTION.                      READ NEXT CH AND
;                                ECHO IT OUT.
; INPUT.                         NONE.
; OUTPUT.                        CH. IN ACC 1, RIGHT
;                                ADJUSTED WITH BIT 0 SET
;                                TO 0.
; CALLS.                         GET73,PUT75.
; CALLED-BY.                     RNCC5,MCD85.
; GLOBAL-VARIABLES-USED.         NONE.
; GLOBAL-VARIABLES-CHANGED.      NONE.
; ERROR-BITS-SET.                NONE.
;
; LOCATIONS FOR SAVING ACC CONTENTS.
;
AC174:   .BLK    1
AC274:   .BLK    1
AC374:   .BLK    1
;
; SAVE ACC CONTENTS.
;
RE174:   STA     1,AC174
         STA     2,AC274
         STA     3,AC374
;
; GET NEXT CHARACTER AND PRINT IT OUT
;
         JSRS    GET73
         JSRS    PUT75
;
; RESTORE ACC CONTENTS AND RETURN.
;
         LDA     1,AC174
         LDA     2,AC274
         LDA     3,AC374
```

```
;
        JMP     0,3
;
;
;*************************************************************
;
; NAME.                          PUT75.
;
; FUNCTION.                      PUTS OUT NEXT CH. ONTO
;                                OUTPUT DEVICE(OUTDV).
;
; INPUT.                         IN ACC 0 RIGHT ADJUSTED.
;
; OUTPUT.                        NONE.
;
; CALLS.                         NONE.
;
; CALLED-BY.                     LOTS OF ROUTINES.
;
; GLOBAL-VARIABLES-USED.         OUTDV.
;
; GLOBAL-VARIABLES-CHANGED.      NONE.
;
; ERROR-BITS-SET.                NONE.
;
;
; LOCATIONS FOR SAVING ACC CONTENTS.
;
AC075:  .BLK    1
AC175:  .BLK    1
DOA75:          61100
SCN75:          63600
;
;     SAVE ACC CONTENTS.
;
PU175:  STA     0,AC075
        STA     1,AC175
;
; SET UP INST. DEP ON DEVICE.
;
        LDA     0,OUTDV
        MOV     0,0,SNR
        JMP     END75
        LDA     1,DOA75
        ADD     0,1
        STA     1,INO75
```

```
;
          LDA     1,SC.75
          ADD     0,1
          STA     1,INT75
;
          LDA     0,AC075
INC75:    .BLK    1
INT75:    .BLK    1
          JMP     .-1
;
;
;  RESTORE ACC CONTENTS AND RETURN.
;
END75:    LDA     0,AC075
          LDA     1,AC175
          JMP     0,3
;
;*****************************************************
;
; NAME.                          GET76.
; FUNCTION.                      GET NEXT CH. FROM
;                                INPUT DEVICE(INPDV)
; INPUT.                         NONE.
; OUTPUT.                        CH. IN ACC 0 RIGHT ADJUSTED
;                                WITH BIT 8 SET TO 0.
; CALLS.                         NONE.
; CALLED-BY.                     REQ74.
; GLOBAL-VARIABLES-USED.         INPDV.
; GLOBAL-VARIABLES-CHANGED.      NONE.
; ERROR-BITS-SET.                NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC176:    .BLK    1
;
```

```
; MASK
;
MSK78:          177
NIO78:          60100
SDN78:          63300
DIA78:          60400
LFD78:          12
;
; SAVE ACC CONTENTS
;
GE178:  STA     1,AC178
        LDA     0,INPDV
        MOV     0,0,SNR
        JMP     END78
        LDA     1,NIO78
        ADD     0,1
        STA     1,IND78
        LDA     1,SDN78
        ADD     0,1
        STA     1,INT78
        LDA     1,DIA78
        ADD     0,1
        STA     1,INF78
;
;
IND78:  .BLK    1
INT78:  .BLK    1
        JMP     .-1
INF78:  .BLK    1
        LDA     1,MSK78
        AND     1,0
        LDA     1,LFD78
        SUBE    0,1,SNR
        JMP     IND78
;
; RESTORE ACC CONTENTS AND RETURN.
;
END78:  LDA     1,AC178
        JMP     0,3
;
;*****************************************************
;
; NAME.                         SST-1.
; FUNCTION.
```

```
;                                SEARCHES SYMBOL TABLE AND
;                                INSERTS .IF ALREADY PRESENT
;                                ERROR BIT IS SET.
; INPUT.                         CHPTR VALUE BETWEEN 2
;                                AND 7.
; OUTPUT.                        NONE.
; CALLS.                         ST379,ERR26.
; CALLED-BY.                     VL842.
; GLOBAL-VARIABLES-USED.         INSLN,CHPTR,PSFLG.
; GLOBAL-VARIABLES-CHANGED.      NONE.
; ERROR-BITS-SET.                BIT NO. 0(I), 1(M).


; SPACE FOR SAVING ACC CONTENTS.

AC051:   .BLK    1
AC151:   .BLK    1
AC251:   .BLK    1
AC351:   .BLK    1

; CONSTANTS.

CN051:           0
CN151:           1
CN251:           2
CN351:           3

MSK51:           77400
CNA51:           40400          ; ^A^
CNB51:           41000          ; ^B^
CNX51:           54000          ; ^X^
CTR51:   .BLK    1

; SPACE FOR SAVING LABEL

LBL51:   .BLK    3
LBA51:           LBL51

;  SAVE ACC CONTENTS.
```

```
;
SS151:  STA     0,AC051
        STA     1,AC151
        STA     2,AC251
        STA     3,AC351
;
; CLEAR LBL51
;
        LDA     2,LBA51
        SUB     0,0
        STA     0,0,2
        STA     0,1,2
        STA     0,2,2
;
; SET UP INDEX REGISTERS AND COUNTER.
;
        LDA     2,INSLN
        LDA     3,LBA51
        LDA     0,CHPTR
        STA     0,CTR51
        DSZ     CTR51
;
; ACC 2 CONTAINS STARTING ADDRESS OF LABEL IN INSLN.
; ACC 3 CONTAINS STARTING ADDRESS OF
; LOCAL SPACE FOR LABEL.
; CTR51 CONTAINS LENGTH OF LABEL.
;
; MOVE LABEL INTO LOCAL SPACE.
;
NXT51:  LDA     0,0,2
        DSZ     CTR51
        JMP     EVN51
;
        LDA     1,MSK51
        ADD     1,0
        STA     0,0,3
        JMP     OKY51
;
;
EVN51:  STA     0,1,3
        INC     3,3
        INC     2,2
        DSZ     CTR51
        JMP     NXT51
        JMP     OKY51
;
```

```
; WHOLE LABEL HAS BEEN TRANSFERRED.LBL51 CONTAINS LABEL
; PADDED ON RIGHT BY NULLS.
; CHECK FOR VALIDITY OF LABEL,
; ( A,B OR X)
;
OKY51:  LDA     0,LBL51
        LDA     1,CNA51                 ; CH A.
        SUBE    0,1,SNR                 ;TEST
        JMP     ERR51                   ; CH. A,ERROR.
;
        LDA     1,CNB51                 ; CH B.
        SUBE    0,1,SNR                 ;TEST
        JMP     ERR51                   ;IT IS CH B.
;
        LDA     1,CNX51                 ; CH X.
        SUBE    0,1,SNR                 ;TEST
        JMP     ERR51                   ; CH. X,ERROR.
;
; LABEL IS OKAY.
;
; CALL ON ROUTINE TO CHECK WHETHER LABEL IS PRESENT
; OR NOT,AND IF NOT PRESENT,INSERT LABEL.
;
        LDA     0,LBA51
        STA     0,AG151                 ; STARTING ADD.
;                                         OF LABEL.
;
        LDA     0,PSFLG
        LDA     1,CN151
        SUB     1,0
        STA     0,AG251                 ;FLAG DENOTING
;                                         WHETHER LABEL
;                                         HAS TO BE
;                                         INSERTED.
        JSRS    ST379
AG151:  .BLK    1
AG251:  .BLK    1
AG351:  .BLK    1
FLG51:  .BLK    1
;
; CHECK RETURNED VALUE TO SEE WHETHER FOUND ,IF AG351
; IS NEGATIVE,NOT FOUND.
;
        LDA     0,FLG51
        IOVLE   0,0,SZC
        JMP     NFD51                   ; NOT FOUND.
;
```

```
;
; FOUND.
;
FND51:  LDA     0,PSFLG
        LDA     1,CN151                 ; CONSTANT 1.
        SUBE    0,1,SZR
        JMP     END51                   ; PASS-2,
;                                        DO NOTHING.
;
; PASS-1,ENTER INTO BIT 1 OF PRETR.
        LDA     0,CN151
        STA     0,AG451
        JSRS    ENE26
AG451:  .BLK    1
        JMP     END51
;
; NOT FOUND IN TABLE.
;
NFD51:  LDA     0,PSFLG
        LDA     1,CN151                 ; CONSTANT 1.
        SUBE    0,1,SNR                 ; TEST FOR PASS.
        JMP     END51                   ; PASS-1,
;                                        DO NOTHING.
;
; PASS IS 2.
;
        JMP     .+1
        JMP     .+1
        JMP     .+1
        JMP     .+1
        JMP     .+1
        JMP     END51
;
; ERROR IN LABEL,ENTER AS BIT 0.
;
ERR51:  SUB     0,0
        STA     0,AG651
        JSRS    ENE26
AG651:  .BLK    1
;
; RESTORE ACC CONTENTS AND RETURN.
;
END51:  LDA     0,AC051
        LDA     1,AC151
        LDA     2,AC251
        JMPS    AC351
;
```

```
;*****************************************************
; NAME.                          HS180.
; FUNCTION.                      PERFORMS THE SHIFTING PART
;                                OF THE HASHING OPERATION
;                                USING THE SYMBOL TABLE
;                                LENGTH AND PRODUCES THE
;                                HASH VALUE IN ACC1.
; INPUT.                         ACC 0.
; OUTPUT.                        ACC 1.
; CALLS.                         NONE.
; CALLED-BY.                     STB79.
; GLOBAL-VARIABLES-USED.         OPFTB.
; GLOBAL-VARIABLES-CHANGED.      NONE.
; ERROR-BITS-SET.                NONE.

; SPACE FOR SAVING ACC CONTENTS.
AC380:  .BLK    1
AC280:  .BLK    1
;
; SAVE ACC CONTENTS.
;
HS180:  STA     3,AC380
        STA     2,AC280
        LDA     3,OPFTB
        LDA     2,3,3           ; LOAD SYMBOL
;                                 TABLE LENGTH.
        SUBZ    1,1             ; CLEAR ACC 1
        MOVZ    0,0             ; CLEAR CARRY.
        MOVZ    2,2
;
;
LP180:  MOVR#   2,2,SZC
        JMP     NXT80
```

```
          ICVL2   0,0
          ICVL    1,1
          JMP     LP18C
NXT8C:    ADD     1,0

; ACC 1 CONTAINS HASH VALUE H(K)
; ACC 0 RETURNS THE CHANGED K.

          LDA     2,AC25C
          JMPS    AC38C

;*****************************************************

; NAME.                          STB79.
; FUNCTION.                      DOES THE ACTUAL SEARCHING
;                                AND INSERTING OPERATIONS
;                                ON THE HASHED TABLE.
; INPUT.                         ADDRESS OF LABEL IN
;                                LOCATION ≠ACC 3 ≠.
;                                FLAG VALUE IN LOCATION
;                                ≠ ACC 3 + 1 ≠.
;                                FLAG=0 DENOTES INSERTION
;                                NECC.
; OUTPUT.                        VALUE IN LOCATION ACC 3 + 2,
;                                NEGATIVE NO. IN LOCATION
;                                ACC 3 + 3 IF NOT FOUND.
;                                RETURNS TO LOC. ACC 3 + 4.
; CALLS.                         HSH80,ORR77,WRT86.
; CALLED-BY.                     SYM95,SST91.
; GLOBAL-VARIABLES-USED.         SYMTB,OPFTB,SLOTD
; GLOBAL-VARIABLES-CHANGED.      SYMT1,ACLBL,NOSYM,OUTOV.
; ERROR-BITS-SET.                NONE.

; CONSTANTS.
```

```
CM.179:             177777
CM.179:             4
CM.379:             1
KEY79:    .BLK      1
VAL79:    .BLK      1
HS279:    .BLK      1
CM.279:             3
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC079:    .BLK      1
AC179:    .BLK      1
AC279:    .BLK      1
AC379:    .BLK      1
; SAVE ACC CONTENTS.
;
ST179:    STA       0,AC079
          STA       1,AC179
          STA       2,AC279
          STA       3,AC379
;
; CLEAR FLAG.
;
          SUB       0,0
          STA       0,3,3
;
; GET A SINGLE WORD BY ORRING.
;
          LDA       2,0,3                ;LOAD ADDRESS.
          LDA       0,0,2
          LDA       1,1,2
;
          STA       0,AG179
          STA       1,AG279
          JSR@      ORR77                ; OR THEM.
AG179:    .BLK      1
AG279:    .BLK      1
AG379:    .BLK      1
;
;
          LDA       0,AG379              ; LOAD RESULT.
          LDA       1,2,2
          STA       0,AG479
          STA       1,AG579
          JSR@      ORR77
AG479:    .BLK      1
```

```
AG579:    .BLK    1
AG679:    .BLK    1
;
; GET HASH VALUE CALCULATED THROUGH
; H(K)=(M(A DIV N  *K) MOD 1)
; WITH A=3,N=LENGTH OF SYMTB,K AS IN AG679,M=16.
;
          LDA     0,AG679                 ; K
          MOV     0,1
          ADC     1,1
          ADC     1,0                     ; MPY BY 3.
;
; ACC 1 HAS AK MOD N.
;
; THE NEXT SEQ OF INSTRUCTIONS EXTRACT THE MSCC BITS
; FROM ACC 0,AND PUTS THEM IN ACC 1.
;
          JSRS    HSH80
;
; ACC 0 CONTAINS THE CHANGED #K#.
; ACC 1 CONTAINS H(K).
; MULTIPLY BY 4 SINCE EACH ENTRY HAS 4 WDS,
;
          MOVLZ   1,1
          MOVLZ   1,1
;
          STA     0,VAL79
          STA     1,KEY79
;
          LDA     3,SYMTB
          ADD     1,3
          MOV     3,2
;
          LDA     0,0,2                   ; LOAD FIRST WD.
          MOV     0,0,SNR                 ; CHK FOR EPY.
          JMP     EPY79
;
; CHECK WHETHER KEY MATCHES.
;
          LDAS    3,AG379
;
          LDA     0,0,3
          LDA     1,0,2
```

```
            SUBE    0,1,SZR                 ; CHECK
            JMP     NCM79
;
            LDA     0,1,3
            LDA     1,1,2
            SUBE    0,1,SZR
            JMP     NCM79                   ; NO MATCH
;
            LDA     0,2,3
            LDA     1,2,2
            SUBE    0,1,SZR                 ; CHECK
            JMP     NCM79                   ;NO MATCH.
;
; ENTRY FOUND.
;
            JMP     FND79
;
; NO MATCH. REHASH.
;
NCM79:      LDA     0,VAL79
            JSRS    HSH80
;
; ACC 1 CONTAINS H(K).
; INTRODUCE 1 IN BIT 0.
;
            LDA     0,ON379
            STA     0,AG779
            STA     1,AG879
            JSRS    ORR77
AG779:      .BLK    1
AG879:      .BLK    1
AG979:      .BLK    1
;
            LDA     1,AG879
            MOVZL   1,1
            MOVZL   1,1
            STA     1,HS279
;
; HS279 CONTAINS VALUE H2(K).
;
NXT79:      LDA     3,OPFTB; LOAD OPFTB ADRESS.
            LDA     2,3,3                   ; LENGTH.
            MOVZL   2,2
            MOVZL   2,2
            LDA     1,HS279
            LDA     0,KEY79
            SUB     1,0                     ;I=I-H2(K)
```

```
          ISZ     0,3,SZC
          ADD     2,3
          STA     3,KEY79                 ; IF SO ADD N.
          MOV     0,3                     ; STORE BACK
;
;;; CHECK IF EMPTY.
;
          LDA     2,SYMTB
          ADC     3,2
          LDA     0,0,2                   ;LOAD FIRST WD.
          MOV     0,0,SNR
          JMP     EPY79                   ; EMPTY ENTRY.
;
;;; CHECK WHETHER KEY MATCHES.
;
          LDAS    3,AC379
;
          LDA     0,0,3
          LDA     1,0,2
          SUBE    0,1,SZR
          JMP     NMT79
;
          LDA     0,1,3
          LDA     1,1,2
          SUBE    0,1,SZR                 ; CHECK
          JMP     NMT79                   ; NO MATCH.
;
          LDA     0,2,3
          LDA     1,2,2
          SUBE    0,1,SZR
          JMP     NMT79                   ;NO MATCH
;
;;; MATCH FOUND,LOAD LOCTR VALUE.
;
          JMP     FND79
;
;;; EMPTY ENTRY,INSERT IF FLAG PERMITS.
;;; ADDRESS IS IN AC 2.
;
EPY79:    LDA     0,CNC79
          STA     2,ADLBL
          LDA     3,AC379
          STA     0,3,3                   ; SET FLAG.
;
;;; CHECK FLAG TO SEE IF INSERTION IS NECC.
;
```

```
          LDA     0,1,3
          LDA     1,CN379                 ; DEC 1
          SUBE    0,1,SNR
          JMP     END79
;
;  INSERT LABEL.
;
;  NOSYM IS A COUNTER FOR CHECKING OVERFLOW.
;
          DSZ     NOSYM
          JMP     .+2
          JMP     ERR79                   ; OVERFLOW.
;
;  INSERT LABEL .
;
;  ACC 2 CONTAINS ADDRESS.
;
          LDAS    3,AC379
          LDA     0,0,3
          STA     0,0,2
          LDA     0,1,3
          STA     0,1,2
          LDA     0,2,3
          STA     0,2,2
          LDA     0,LOCTR
          STA     0,3,2
          JMP     END79
;
;  ENTRY FOUND .SET VALUE FOR LOCTR.
;
END79:    LDA     0,3,2
          STA     2,ADLBL
          LDA     3,AC379
          STA     0,2,3
;
;  RESTORE ACC CONTENTS AND RETURN
;
END79:    LDA     0,AC079
          LDA     1,AC179
          LDA     2,AC279
          LDA     3,AC379
;
          JMP     4,3
;
;  ERROR ,OVERFLOW IN TABLE.
;
MES79:    .TXT    /<012><015>OVERFLOW,ABORT/
```

```
MEA79:            MES79
ERR79:   LDA      2,MEA79
;
         LDA      0,SLOTO
         STA      0,OUTOV
;
         JSR$     WRT65
         HALT
;*****************************************************************
;
; NAME.
;                                      ORR77.
; FUNCTION.
;                                      PERFORMS THE LOGICAL
;                                      OR OPERATION ON INPUTS.
; INPUT.
;                                      LOCATIONS ≠AC 3≠ ≠AC 3 + 1≠
; OUTPUT.
;                                      LOCATION ≠AC 3 + 2≠.
; CALLS.
;                                      NONE.
; CALLED-BY.
;                                      STB79,ERE26,DER14.
; GLOBAL-VARIABLES-USED.
;                                      NONE.
; GLOBAL-VARIABLES-CHANGED.
;                                      NONE.
; ERROR-BITS-SET.
;                                      NONE.
;
;
;
; LOCATIONS FOR SAVING ACC CONTENTS.
;
AC077:   .BLK     1
AC177:   .BLK     1
AC277:   .BLK     1
AC377:   .BLK     1
;
; SAVE ACC CONTENTS.
;
OR177:   STA      0,AC077
         STA      1,AC177
         STA      2,AC277
         STA      3,AC377
;
; LOAD IN AND PERFORM OR FUNCTION.
```

```
;
        LDA     0,0,3
        LDA     1,1,3
        COM     0,0
        COM     1,1
        AND     0,1
        COM     1,1
        STA     1,2,3
;
; RESTORE ACC AND RETURN.
;
        LDA     0,AC077
        LDA     1,AC177
        LDA     2,AC277
        LDA     3,AC377
;
        JMP     3,3
;
;*****************************************************
;
; NAME.                             UPC11.
; FUNCTION.                         UPDATES LINCTR,LOCTR,LOCTN.
; INPUT.                            NONE.
; OUTPUT.                           NONE.
; CALLS.                            NONE.
; CALLED-BY.                        PASS1,PASS2.
; GLOBAL-VARIABLES-USED.            LENTH.
; GLOBAL-VARIABLES-CHANGED.         LOCTN,LOCTR,LINCTR.
; ERROR-BITS-SET.                   NONE.
;
;
;   SPACE FOR SAVING ACC CONTENTS.
;
AC011:  .BLK    1
AC111:  .BLK    1
;
```

```
CNG11:          1
;
;  SAVE ACC CONTENTS.
;
UP111:  STA     0,ACC011
        STA     1,AC111
;
;  UPDATE COUNTERS.
;
        LDA     0,LNCTR
        LDA     1,CNG11
        ADD     1,0
        STA     0,LNCTR
;
        LDA     0,LOCTR
        LDA     1,LENTH
        ADD     1,0
        STA     0,LOCTR
        STA     0,LOCTN
;
;  RESTORE ACC CONTENTS AND RETURN.
;
        LDA     0,ACC011
        LDA     1,AC111
;
        JMP     0,3
;
;*****************************************************
;
; NAME.                         ENETS.
; FUNCTION.                     ENTER PRERR INTO ERROR
;                               TABLE.
; INPUT.                        NONE.
; OUTPUT.                       NONE.
; CALLS.                        NONE.
; CALLED-BY.                    PASS1,PASS2.
; GLOBAL-VARIABLES-USED.        PRERR,ER1TB,ER2TB,PSFLG.
; GLOBAL-VARIABLES-CHANGED.
```

```
;
;  ERROR-BITS-SET.                    ER1T1,ER2T1,E1EPT,E2EPT.
;                                     NONE.
;
;
;  SPACE FOR SAVING ACC CONTENTS.
;
AC0C09:    .BLK     1
AC1C09:    .BLK     1
AC2C09:    .BLK     1
AC3C09:    .BLK     1
;
;  CONSTANTS.
;
CN0C09:             62                          ; DEC 50
CN1C09:             1
;
;  SAVE ACC CONTENTS,ENTRY.
;
EN1C09:    STA      0,AC0C09
           STA      1,AC1C09
           STA      2,AC2C09
           STA      3,AC3C09
;
;  CHECK WHETHER PRERR IS ZERO.IF SO ,RETURN.
;
           LDA      0,PRERR
           MOV      0,0,SNR
           JMP      END09
;
;  CHECK WHICH PASS AND SET ACC 3,ACC 2,TO
;  ER1TB OR ER2TB AND E1EPT OR E2EPT ACCORDINGLY.
;
           LDA      0,PSFLG
           LDA      1,CN1C09
           SUBE     0,1,SZR
           JMP      PS209
;
;   PASS IS 1.SET CORR. VALUES.
;
           LDA      3,ER1TB
           LDA      2,E1EPT
           JMP      NXT09
;
;  PASS IS 2.SET CORR. VALUES.
;
```

```
PS2C9:  LDA     3,ER2TB
        LDA     2,E2EPT
;
; CHECK WHETHER TABLE IS FULL.IF SO , RETURN.
;
NXTC9:  LDA     0,CN009                 ; DEC 50.
        MOV     3,1
        ADC     0,1
        SUBB    1,2,SNR
        JMP     ENC09
;
; TABLE NOT FULL,ENTER ERROR NO.
;
        LDA     0,ENCTR
        STA     0,0,3
        LDA     0,PSERR
        STA     0,1,2
;
;  UPDATE E1EPT OR E2EPT.
;
        INC     2,2
        INC     2,2
;
; REPLACE DEPENDING ON PASS.
;
        LDA     0,PSFLG
        LDA     1,CN109                 ; DEC   1
        SUBB    0,1,SZR
        JMP     P2EC9                   ; PASS IS 2.
;
; PASS IS 1 . REPLACE AS E1EPT.
;
        STA     2,E1EPT
        JMP     ENC09
;
; PASS IS 2.REPLACE AS E2EPT.
;
P2EC9:  STA     2,E2EPT
;
;  RESTORE ACC CONTENTS AND RETURN.
;
ENC09:  LDA     0,AC009
        LDA     1,AC109
        LDA     2,AC209
        JMPS    AC309
;
```

```
;*****************************************************
;
; NAME.                          GLB41.
; FUNCTION.                      SET CHPTR TO THE FIRST
;                                BLANK CHARACTER IN ITSLN
;                                AFTER THE CURRENT VALUE.
; INPUT.                         NONE.
; OUTPUT.                        CHPTR POINTS TO BLANK CH.
; CALLS.                         GCHb1.
; CALLED-BY.                     LAC21.
; GLOBAL-VARIABLES-USED.         NONE.
; GLOBAL-VARIABLES-CHANGED.      CHPTR.
; ERROR-BITS-SET.                NONE.
;
; SPACE  FOR SAVING AC
; SPACE  FOR SAVING ACC CONTENTS.
;
AC041:   .BLK    1
AC141:   .BLK    1
AC241:   .BLK    1
AC341:   .BLK    1
;
;  CONSTANTS.
;
CH.041:          40                    ;  BLANK
;
; SAVE ACC CONTENTS,ENTRY POINT.
;
GL141:   STA     0,AC041
         STA     1,AC141
         STA     2,AC241
         STA     3,AC341
;
; GET NEXT WORD.
;
NXT41:   JSRS    GCHb1
; CHECK WHETHER BLANK OR NOT.
         LDA     1,CH.041              ; BLANK.
```

```
        SUBE    0,1,SNR
        JMP     FND41

; CH. IS NOT BLANK,GET NEXT CHARACTER.

        ISZ     CHPTR
        JMP     NXT41

; SEPARATOR FOUND,RESTORE ACC CONTENTS AND RETURN.

FND41:  LDA     0,AC041
        LDA     1,AC141
        LDA     2,AC241
        JMPS    AC341

;*****************************************************

; NAME.                         SCP23.
; FUNCTION.                     SETS CHPTR TO NEXT
                                NON-BLANK CHARACTER.
; INPUT.                        NONE.
; OUTPUT.                       CHPTR POINTING TO NON-BLANK
                                CHARACTER.
                                TAKES  # ACC 3 # RETURN
                                IF ERROR.
                                TAKES #ACC 3 + 1# RETURN
                                OTHERWISE.
; CALLS.                        ENE26,SCM=1.
; CALLED-BY.                    LOTS.
; GLOBAL-VARIABLES-USED.        NONE.
; GLOBAL-VARIABLES-CHANGED.     CHPTR.
; ERROR-BITS-SET.               BIT NO. 10 (F).

; SPACE FOR SAVING ACC CONTENTS.
ACC23:  .BLK    1
```

```
AC123:    .BLK     1
AC223:    .BLK     1
AC323:    .BLK     1
;
; CONSTANTS.
;
CN123:             40                          ; BLANK
CN223:             120                         ; DEC '80
CTR23:    .BLK     1
TEN23:             12
;
; SAVE ACS CONTENTS.
;
SC123:    STA      0,AC023
          STA      1,AC123
          STA      2,AC223
          INC      3,3
          STA      3,AC323
;
; SET COUNTER TO COUNT UPTO END OF LINE.
;
          LDA      0,CHPTR
          LDA      1,CN223
          SUB      0,1
          INC      1,1
          STA      1,CTR23
;
; GET NEXT CH. AND CHECK.
;
NXT23:    JSRS     GCH61
;
; CHECK OUT CHARACTER.
;
          LDA      1,CN123                     ; BLANK.
          SUBE     0,1,SNR
          JMP      BLK23
          JMP      FND23
;
; CH. IS A BLANK,GET NEXT CH.
;
BLK23:    ISZ      CHPTR
          DSZ      CTR23
          JMP      NXT23
;
; ENTER ERROR BIT NO. 10.
; (NO NON-BLANK CH. FOUND).
;
```

```
        LDA     0,TEM23
        STA     0,ARG23
        JSR@    ENI26
ARG23:  .BLK    1
        DSZ     AC323
;
; CH. IS NOT BLANK.
; RESTORE ACCS AND RETURN.
;
END23:  LDA     0,AC023
        LDA     1,AC123
        LDA     2,AC223
        JMP@    AC323
;
;
;***********************************************************
;
; NAME.                          PAOP7.
; FUNCTION.                      PROCESS OP-CODE,OPERANDS.
;                                IN PASS 1 ,GET LENTH.
;                                IN PASS 2 ,GET LENTH
;                                AND SET WSPCE(ASSEMBLE WD).
; INPUT.                         NONE.
; OUTPUT.                        NONE.
; CALLS.                         SOP23,VTR24,SPM25.
; CALLED-BY.                     PASS-1,PASS-2
; GLOBAL-VARIABLES-USED.         ABFLG.
; GLOBAL-VARIABLES-CHANGED.      CHBEG.
; ERROR-BITS-SET.                NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC007:  .BLK    1
AC107:  .BLK    1
AC207:  .BLK    1
AC307:  .BLK    1
;
```

```
; SAVE ACC CONTENTS,ENTRY POINT.
;
PR107:  STA     0,AC007
        STA     1,AC107
        STA     2,AC207
        STA     3,AC307
;
; SET CHPTR.
;
        JSR$    SCP23
        JMP     END07                   ; ERROR RETURN.
;
; LINE IS OKAY,CHECK VALID TERMINATOR.
;
        JSR$    VTR24
;
; CHECK ASFKG TO SEE IF LINE IS QUESTIONABLE.
;
        LDA     0,ASFLG
        MOV     0,0,SNR
        JMP     END07
;
; LINE IS OKAY,SEARCH TABLES AND SERVICE,.
;
        JSR$    SR125
;
; SET CHBEG AND RESTORE ACCS,RETURN.
;
END07:  LDA     0,CHPTR
        STA     0,CHBEG
        LDA     0,AC007
        LDA     1,AC107
        LDA     2,AC207
        JMP$    AC307
;
;*****************************************************
;
; NAME.                         GCH31.
; FUNCTION.                     GETS CHARACTER POINTED AT
;                               BY CHPTR FROM INSLN.
; INPUT.                        NONE.
; OUTPUT.                       IN ACC 0,RIGHT  ADJUSTED.
```

```
; CALLS.                          NONE.
; CALLED-BY.                      LOTS.
; GLOBAL-VARIABLES-USED.          CHPTR,INSLN.
; GLOBAL-VARIABLES-CHANGED.       NONE.
; ERROR-BITS-SET.                 NONE.

; SPACE FOR SAVING ACC CONTENTS.

AC161:  .BLK    1
AC261:  .BLK    1
AC361:  .BLK    1

; CONSTANTS.

MSK61:          377
CN161:          1
CHK61:          121
BLK61:          40

; ENTRY POINT,SAVE ACC CONTENTS.

GC161:  STA     1,AC161
        STA     2,AC261
        STA     3,AC361

; CHECK CHPTR GREATER THAN 81

        LDA     0,CHPTR
        LDA     1,CHK61
        SUBLE   1,0,SNC
        JMP     ERR61

; FIND OUT WHICH WORD AND WHICH HALF.

        LDA     0,CHPTR
        LDA     1,CN161
        SUB     1,0
        MOVZR   0,0,SZC

; CARRY IS NON-ZERO,SECOND HALF TO BE LOADED.
```

```
          JMP     NZC81

;  CARRY IS ZERO,FIRST  IS NEED.
;  ACC 0 CONTAINS THE WD TO BE LOADED.

          LDA     2,INSLN
          ADC     0,2
          LDA     0,0,2
          MOVS    0,0
          LDA     1,MSK81
          ANC     1,0

;  ACC 0 CONTAINS REQD. WD.

          JMP     END81

;  CARRY IS NON-ZERO.SECOND HALF TO BE
;  LOADED.ACC 0 CONTAINS WORD TO BE LOADED.

NZC81:    LDA     2,INSLN
          ADC     0,2
          LDA     0,0,2
          LDA     1,MSK81
          ANC     1,0
          JMP     END81

;  CHPTR GREATER THAN 80.
;  SEND BACK BLANK.
;  SET CHPTR TO 81.

ERR81:    LDA     0,CHK81
          STA     0,CHPTR
          LDA     0,BLK81

;  ACC 0 CONTAINS REQD, WD,

END81:    LDA     1,AC181
          LDA     2,AC281
          JMPS    AC381

;*****************************************************

;  NAME.                    VTR24.
;  FUNCTION.                CHECK VALIDITY OF OP-CODE
```

```
;                                       TERMINATOR .I.E. CHECK IF
;                                       IT IS ≠A ≠,≠B ≠,≠ ≠.
;  INPUT.
;                                       CHPTR POINTING TO FIRST
;                                       CHARACTER IN OP-CODE.
;  OUTPUT.
;                                       ABFLG SET TO 0,1,2, OR 3.
;                                       0 IF INVALID.
;                                       1 IF ≠A ≠.
;                                       2 IF ≠B ≠.
;                                       3 IF ≠ ≠.
;                                       CHPTR POINTS TO FIRST CH.
;                                       OF OP-CODE.
;  CALLS.
;                                       ENE86,GCH81.
;  CALLED-BY.
;                                       PROA7.
;  GLOBAL-VARIABLES-USED.
;                                       NONE.
;  GLOBAL-VARIABLES-CHANGED.
;                                       ABFLG,CHPTR.
;  ERROR-BITS-SET.
;                                       10(F),13(Q).

;  SPACE FOR SAVING ACC CONTENTS.

AC024:    .BLK    1
AC124:    .BLK    1
AC224:    .BLK    1
AC324:    .BLK    1

;  CONSTANTS.

BLK24:            40                    ;  BLANK
CHA24:            101
CHB24:            102
CNC24:            12
CN124:            120                    ; DEC 80
CN324:            1
CN424:            2
CN524:            15                    ; DEC. 13.
THR24:            3

;  SAVE ACC CONTENTS.

VT124:    STA     0,AC024
```

```
        STA     1,AC124
        STA     2,AC224
        STA     3,AC324
;
; INCREMENT CHPTR TO POINT TO CH AFTER OPCODE.
;
        ISZ     CHPTR
        ISZ     CHPTR
        ISZ     CHPTR
;
; CHECK IF CHPTR GREATER THAN 80.
;
        LDA     0,CHPTR
        LDA     1,CH124                 ; DEC 80
        SUBLE   0,1,SZC
;
; CARRY IS NON-ZERO,LINE IS QUESTIONABLE.
;
        JMP     QN324
;
; LINE IS OKAY,CHECK CH. TO SEE IF IT IS BLANK.
;
        JSRS    GC161                   ; GET CHARACTER.
        LDA     1,BLK24
        SUBE    0,1,SNR
        JMP     OKY24
;
;
        LDA     1,CHA24
        SUBE    0,1,SNR
        JMP     CKA24
;
        LDA     1,CHB24
        SUBE    0,1,SNR
        JMP     CKB24
;
;
        JMP     NTR24
;
; CH. IS A,SET ABFLG TO 1.
;
CKA24:  LDA     0,CH324                 ; DEC  1.
        STA     0,ABFLG
        JMP     CHK24
;
; CH. IS B,SET ABFLG TO 2.
```

```
;
CKB24:  LDA     0,CN424                 ; DEC.  2
        STA     0,ABFLG
        JMP     CHK24
;
; CHARACTER IS A #,SET ABFLG  TO 5,RETURN.
;
CKY24:  LDA     0,TMB24
        STA     0,ABFLG
        JMP     LB224
;
; CH. IS A OR B.CHECK NEXT CHARACTER.
;
CHK24:  ISZ     CHPTR
        JSRS    GCH161
;
; ACC 0 CONTAINS CHARACTER.
;
        LDA     1,BLK24
        SUBZ    0,1,SNR
        JMP     LB124
;
;
; INVALID TERMINATOR.
; SET BIT NO. 10(F) OF ERROR WORD.
; SET ABFLG TO 0.
;
NTR24:  LDA     0,CN624                 ; DEC. 16.
        STA     0,AG124
        JSRS    ENE26
AG124:  .BLK    1
        SUB     0,0
        STA     0,ABFLG
        JMP     END24
;
; LINE IS QUESTIONABLE.
; SET BIT NO.13(Q).
; SET ABFLG TO 0.
;
CNB24:  SUB     0,0
        STA     0,ABFLG
        LDA     0,CN524
        STA     0,AG224
        JSRS    ENE26
AG224:  .BLK    1
        JMP     END24
;
```

```
; SET CHPTR TO POINT TO BEG OF OPCODE.
;
LB124:  DSZ     CHPTR
LB224:  DSZ     CHPTR
        DSZ     CHPTR
        DSZ     CHPTR
;
; RESTORE ACC CONTENTS AND RETURN.
;
END24:  LDA     0,AC024
        LDA     1,AC124
        LDA     2,AC224
        JMPS    AC324
;
;*****************************************************
;
; NAME.                         SPM25.
; FUNCTION.                     SEARCHES POT,NOT,
;                               AND IF FOUND TRANSFERS TO
;                               SERVICE ROUTINE.
; INPUT.                        CHPTR POINTS TO FIRST CH.
;                               IN OP-CODE.
; OUTPUT.                       LENGTH,MCPOZ SET.
; CALLS.                        END25,SCH21,CPT43,SPT44,
;                               SERVICE ROUTINES.
; CALLED-BY.                    PROPY.
; GLOBAL-VARIABLES-USED.        NONE.
; GLOBAL-VARIABLES-CHANGED.     CHPTR.
; ERROR-BITS-SET.               4 (0).
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC025:  .BLK    1
AC125:  .BLK    1
AC225:  .BLK    1
AC325:  .BLK    1
```

```
;
;; CONSTANTS.
;
CNG25:          4
;
; SPACE FOR SAVING OPCODE.
;
OP125:   .BLK    1
OP225:   .BLK    1
OPA25:          OP125
;
; SAVE ACC CONTENTS,ENTRY.
;
SP125:   STA     0,AC025
         STA     1,AC125
         STA     2,AC225
         STA     3,AC325
;
; SAVE OPCODE.
;
; CHPTR POINTS TO FIRST CH. GET CHARACTER.
;
         JSRS    GCH81
;
; CHARACTER IS IN ACC 0.
;
         MOVS    0,1
         ISZ     CHPTR
         JSRS    GCH81
         ADD     0,1
         STA     1,OP125                 ; FIRST WD.
         ISZ     CHPTR
         JSRS    GCH81
         MOVS    0,1
         STA     0,OP225                 ; SECOND WD.
; SET CHPTR TO POINT TO END OF OP-CODE.
         ISZ     CHPTR
;
;  CALL OI ROUTINE TO SEARCH POT.
;
         LDA     0,OPA25
         STA     0,AG125
         JSRS    SPT43
AG125:   .BLK    1
AG225:          0
         LDA     0,AG225
;
```

```
;
; TEST WHETHER PRESENT IN POT.
;
        MOV     0,0,SZR
;
; IS PRESENT.TRANSFER TO ROUTINE.
;
        JMP     PRS25
;
; NOT PRESENT,SEARCH NOT.
;
        LDA     0,OPA25
        STA     0,AG325
        JSRS    SMT44
AG325:  .BLK    1
AG425:          0
;
; TEST WHETHER PRESENT IN NOT.
;
        LDA     0,AG425
        MOV     0,0,SZR
;
; RESULT IS NON ZERO,OPCODE IS PRESENT.
;
        JMP     PRS25
;
; ERROR,NOT PRESENT IN EITHER TABLES.
;
;  ENTER ERROR BIT NO. 4
;
        LDA     0,CNJ25                 ; DEC  4
        STA     0,AG525
        JSRS    ERR25
AG525:  .BLK    1
        JMP     END25
;
; ACC CONTAINS SERVICE  ROUTINE ADDRESS.
;
PRS25:  MOV     0,2
        JSR     0,2
;
; RESTORE ACC CONTENTS AND RETURN.
;
END25:  LDA     0,AC025
        LDA     1,AC125
        LDA     2,AC225
        JMPS    AC325
;
```

```
;
;*****************************************************
;
; NAME.
;                               SP143.
; FUNCTION.
;                               SEARCHES THE P.O.T.
; INPUT.
;                               STARTING ADDRESS OF LOCAL
;                               SPACE WHERE OP-CODE IS
;                               STORED(IN ACC. ≠AC 3 ≠.)
; OUTPUT.
;                               SERVICE ROUTINE ADDRESS IF
;                               PRESENT. 0 OTHERWISE(IN ACC
;                               ≠AC 3 + 1).
;                               RETURNS TO ADD. ≠AC 3 + 2≠.
; CALLS.
;                               NONE.
; CALLED-BY.
;                               SP025.
; GLOBAL-VARIABLES-USED.
;                               POEPT,POPTB.
; GLOBAL-VARIABLES-CHANGED.
;                               NONE.
; ERROR-BITS-SET.
;                               NONE.
;
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC043:   .BLK     1
AC143:   .BLK     1
AC243:   .BLK     1
AC343:   .BLK     1
;
; CONSTANT.
;
CN043:           0
;
; TEMPORARY LOCATIONS.
;
POP43:   .BLK     1
POE43:   .BLK     1
;
; SAVE ACC CONTENTS,ENTRY POINT.
;
SP143:   STA      0,AC043
```

```
          STA     1,AC143
          STA     2,AC243
          STA     3,AC343
;
;;;;  SEARCH TABLE FOR MATCH.
;;;;  STORE POPTS,POEPT IN TEMPORARY LOCATIONS.
;;;;
          LDA     0,POEPT
          STA     0,POE43
          LDA     0,POPTS
          STA     0,POP43
;;;;
LOP43:    LDA     0,@POE43
          LDA     2,POP43
          SUBE    0,2,SNR
          JMP     NFD43                   ;WHOLE TABLE HAS
;                                         ; BEEN SEARCHED.
;;;;  COMPARE.
;;;;
          LDA     1,0,2
          LDA     3,AC343
          LDA     0,0,3
          SUBE    0,1,SZR
          JMP     NMT43                   ; NO MATCH
;
;;;;  FIRST WD MATCHES TRY SECOND WD.
;;;;
          LDA     1,1,2
          LDA     0,1,3
          SUBE    0,1,SZR
          JMP     NMT43                   ; NO MATCH
;
;;;;  MATCH FOUND,LOAD SERVICE ROUTINE ADDRESS.
;;;;
          LDA     0,2,2
          LDA     3,AC343
          STA     0,1,3
          JMP     END43
;
;;;;  NO MATCH .TRY NEXT ENTRY.
;;;;
NMT43:    LDA     0,POP43
          INC     0,0
          INC     0,0
          INC     0,0
```

```
          STA     0,ACD43
          JMP     LCP43
;
; NOT FOUND,SET RETURNING VALUE TO -1.
;
NFD43:    LDA     0,ONE43
          LDA     3,AC343
          STA     0,1,3
;
; RESTORE ACC CONTENTS AND RETURN.
;
END43:    LDA     0,ACD43
          LDA     1,AC143
          LDA     2,AC243
          LDA     3,AC343
;
          JMP     2,3
;
;*****************************************************
;
; NAME.                         PRT32.
; FUNCTION.                     PRINTS OUT THE OUTPUT
;                               LINE,AFTER DELETING TRAILIN
;                               BLANKS.
; INPUT.                        NONE.
; OUTPUT.                       NONE.
; CALLS.                        PUT79.
; CALLED-BY.                    PSC10,NER17,PAS60.
; GLOBAL-VARIABLES-USED.        SCFLG,SLOTD
; GLOBAL-VARIABLES-CHANGED.     OUTDV.
; ERROR-BITS-SET.               NONE.
;
; SPACE FOR ACC CONTENTS.
;
ACD32:    .BLK    1
AC132:    .BLK    1
```

```
AC282:  .BLK    1
AC382:  .BLK    1
;
; CONSTANTS AND COUNTERS.
;
CN082:          74                      ; DEC 60
CTR82:  .BLK    1
CN282:          15
CN382:          12
BLK82:          2000
CT182:  .BLK    1
;
; SAVE ACC CONTENTS,ENTRY.
;
PR182:  STA     0,AC082
        STA     1,AC182
        STA     2,AC282
        STA     3,AC382
;
; CHECK IF NECC.
;
        LDA     0,SOFLG
        MOV     0,0,SNR
        JMP     EN082
;
; SET DEVICE CODE.
;
        LDA     0,SLOTC
        STA     0,OUTOV
;
; PRINT OUT CR,LF.
;
        LDA     0,CN282
        JSR@    PC175
        LDA     0,CN382
        JSR@    PC175
;
; SET UP COUNTER TO COUNT UP TO 60 WORDS.
;
        LDA     0,CN382
        STA     0,CTR82
;
; PRINT OUT.
;
        LDA     2,PRTLN
        LDA     0,CN082
        ADD     0,2
```

```
            STA     2,CT1B2
            DSZ     CT1B2
;
;
LBLB2:      LDAS    0,CT1B2
            FDDA    1,BLK32
            SUBBE   0,1,SZR
            JMP     OKY82
            DSZ     CT1B2
            DSZ     CT3B2
            JMP     LBLB2
            JMP     ENDB2
OKY82:      JMP     .+1
            LDA     2,PRTLN
;
;
;
; PRINT OUT NEXT WORD.
;
LOPB2:      LDA     0,0,2
            MOVS    0,0
            JSRS    PUT75
;
            LDA     0,0,2
            JSRS    PUT75
;
; CHECK WHETHER ALL CHARACTERS HAVE BEEN PRINTED OUT.
;
            INC     2,2
            DSZ     CTRB2
            JMP     LOPB2
;
;
; PRINT OUT CR,LF.
;
ENDB2:      LDA     0,CN2B2
            JSRS    PUT75
            LDA     0,CN3B2
            JSRS    PUT75
;
; RESTORE ACCS,RETURN.
;
            LDA     0,AC0B2
            LDA     1,AC1B2
            LDA     2,AC2B2
            JMPS    AC3B2
;
```

```
;
;*************************************************************
;
; NAME.
;                                  SMT44.
; FUNCTION.
;                                  SEARCHES M.O.T.
; INPUT.
;                                  STARTING ADDRESS OF LOCAL
;                                  SPACE WHERE OP-CODE IS
;                                  STORED(IN ADD.≠AC 3 ≠).
; OUTPUT.
;                                  SERVICE ROUTINE ADDRESS IF
;                                  PRESENT.0 OTHRWISE.(IN ADD.
;                                  ≠AC 3 + 1).
;                                  RETURNS TO ADDRESS AC 3 + 2
; CALLS.
;                                  NONE.
; CALLED-BY.
;                                  SP-25.
; GLOBAL-VARIABLES-USED.
;                                  NEACT,OPTB.
; GLOBAL-VARIABLES-CHANGED.
;                                  BSVAL.
; ERROR-BITS-SET.
;                                  NONE.
;
;
; CONSTANTS.COUNTERS.
;
SEF44:                  AC063
LNF44:     .BLK         1
UPF44:     .BLK         1
IND44:     .BLK         1
KEY44:     .BLK         1
MOP44:     .BLK         1
MSK44:                  377                        ; MASK
CNC44:                  0
CN144:                  1
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC044:     .BLK           1
AC144:     .BLK           1
AC244:     .BLK           1
AC344:     .BLK         1
;
```

```
;
; SAVE ACS CONTENTS,ENTRY POINT.
;
SM144:  STA     0,AC044
        STA     1,AC144
        STA     2,AC244
        STA     3,AC344
;
; INITIALISE LWR44,UPR44,KEY44,IND44,MOP44
;
        LDA     0,J,3
        STA     0,KEY44
        LDA     0,REMOT
        STA     0,UPR44
        LDA     0,CN144
        STA     0,LWR44
        LDA     0,CNC44
        STA     0,IND44
        LDA     0,MOPTB
        STA     0,MOP44
;
; CHECK IF UPR44 IS LESS THAN LWR44.
; IF SO,SEARCH HAS ENDED UNSUCCESSFULLY.
;
L4244:  LDA     0,LWR44
        LDA     1,UPR44
        SUBLE   0,1,SZC
        JMP     NF044                   ;UNSUCCESSFULL
;
; SET INDEX (IND44) TO  LOWER BOUND OF (LWR44+UPR44)/2.
;
        ADD     1,0
        MOVZR   0,0
        STA     0,IND44
;
; LOAD ADDRESS OF ENTRY COUNT TO INDEX.
;
        LDA     0,IND44
        LDA     1,CN144
        SUB     1,0
        MOV     0,1
        ADDZ    0,1
        ADDZ    0,1                     ; MULTIPLY BY 3.
        LDA     3,MOP44                 ;LOAD STARTING
;                                        ADDRESS OF MOT.
        ADD     1,3                     ;ADD(3*INDEX).
        LDA     2,KEY44
;
```

```
;AC3 CONTAINS ADDRESS OF ENTRY.
;AC2 CONTAINS ADDRESS OF KEY.
;CHECK FIRST WD.
;
        LDA     0,,,3
        LDA     1,,,2
        SUBE    0,1,SZR
        JMP     NMT44
;
;CHECK SECOND WD.
;
        LDA     0,1,3
        LDA     1,1,2
        SUBE    0,1,SNR
        JMP     FND44
;
;NO MATCH,GET NEXT ENTRY TO BE CHECKED.
;AC0 CONTAINS  MISMATCHED WD OF ENTRY.
;AC1 CONTAINS MISMATCHED WD OF KEY.
;
;CHECK WHETHER K LESS THAN K(I).
;
NMT44:  SUBLE   0,1,SNC
        JMP     LB544                   ;K GREATER
                                         THAN K(I).
;
;K LESS THAN K(I).SET UPR44 = IND44-1
;
        LDA     0,IND44
        LDA     1,CN144
        SUB     1,0
        STA     0,UPR44
        JMP     LB244
;
;K GREATER THAN K(I),SET LWR44=IND44+1
;
LB544:  LDA     0,IND44
        INC     0,0
        STA     0,LWR44
        JMP     LB244
;
;MATCH NOT FOUND,SET RETURNING VALUE TO 0.
;
NFD44:  LDA     0,CN044
        LDA     3,AC344
        STA     0,1,3
```

```
          JMP     END44
;
; ENTRY FOUND.AC 3 CONTAINS ADD. OF MATCHING ENTRY.
; LEFT HALF OF VALUE CONTAINS FORMAT NO.
; RIGHT HALF CONTAINS BASE VALUE.
;
; ISOLATE BASE VALUE.
;
FND44:    LDA     0,2,3                   ; LOAD VALUE
          LDA     1,MSK44                 ; LOAD MASK
          AND     1,0
          STA     0,BSVAL
;
; ISOLATE FORMAT NO.
;
          LDA     0,2,3
          LDA     1,MSK44
          MOVS    0,0
          AND     1,0
;
; LOAD SERVICE ROUTINE ADDRESS.
;
          LDA     3,SER44
          ADD     0,3
          LDA     1,0,3
;
; SET RETURNING VALUE.
;
          LDA     3,AC344
          STA     1,1,3
;
; RESTORE ACC CONTENTS AND RETURN.
;
END44:    LDA     0,AC044
          LDA     1,AC144
          LDA     2,AC244
          LDA     3,AC344
;
          JMP     2,3
;
;*****************************************************
;
; NAME.                         PSC10.
; FUNCTION.                     PRINT OUT LISTING.
```

```
;  INPUT.                          NONE.
;  OUTPUT.                         NONE.
;  CALLS.                          ERL28,ELC29,LOC30,EWS31,
;                                  ESC32,PVR34,PRT62.
;  CALLED-BY.                      PASS1,PASS2.
;  GLOBAL-VARIABLES-USED.          LTFLG,PAGE#.
;  GLOBAL-VARIABLES-CHANGED.       NONE.
;  ERROR-BITS-SET.                 NONE.
;
;
;  SPACE FOR SAVING ACC CONTENTS.
;
AC010:    .BLK      1
AC110:    .BLK      1
AC210:    .BLK      1
AC310:    .BLK      1
;
;  SAVE ACC CONTENTS,ENTRY POINT.
;
PS110:    STA       0,AC010
          STA       1,AC110
          STA       2,AC210
          STA       3,AC310
;
;  CHECK WHETHER LISTING IS NEED.
;
          LDA       0,LTFLG
          MOV       0,0,SNR
          JMP       LNN10
;
;  LISTING IS NEED.
;
          JSR@      ERL28
          JSR@      ELC29
          JSR@      LOC30
          JSR@      EWS31
          JSR@      ESC32
          JSR@      PRT62
```

```
          JSRS    FVR34
          JMP     END10
;
; CHECK WHETHER ERROR-LINE.
;
LNN10:    LDA     0,PRERR
          MOV     0,0,SNR
          JMP     END10
;
; IT IS AN ERROR-LINE.
;
          JSRS    ERL28
          JSRS    ELC29
          JSRS    ERS31
          JSRS    ESC32
          JSRS    PRT82
;
; RESTORE ACCS AND RETURN.
;
END10:    LDA     0,AC010
          LDA     1,AC110
          LDA     2,AC210
          JMPS    AC310
;
;*****************************************************
;
; NAME.                          ELC29.
; FUNCTION.                      ENTERS LINE COUNTER
;                                INTO PRTLN AFTER CONVERSION
;                                INTO DECIMAL INTO
;                                POSITIONS 6-10.
; INPUT.                         NONE.
; OUTPUT.                        NONE.
; CALLS.                         BCC46.
; CALLED-BY.                     PSC10.
; GLOBAL-VARIABLES-USED.         LNCTR.
; GLOBAL-VARIABLES-CHANGED.      PRPTR.
```

```
; ERROR-BIT-SET.                    NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC029:    .BLK    1
AC129:    .BLK    1
AC229:    .BLK    1
AC329:    .BLK    1
;
; CONSTANTS.
;
CHR29:            6
;
; SAVE ACC CONTENTS,ENTRY.
;
ENT29:    STA     0,AC029
          STA     1,AC129
          STA     2,AC229
          STA     3,AC329
;
; CONVERT INTO DECIMAL AND ENTER.
;
          LDA     0,LNCTR
          LDA     1,CHR29
          STA     1,PRPT
          JSRS    BCC46
;
; RESTORE ACC CONTENTS AND RETURN.
;
          LDA     0,AC029
          LDA     1,AC129
          LDA     2,AC229
          JMPS    AC329
;
;*****************************************************
;
; NAME.                             BHX47.
; FUNCTION.                         CONVERTS WORD INTO 4
;                                   HEX CHARACTERS.
; INPUT.                            ACC 0.
```

```
; OUTPUT.                          2 WORDS IN LOCATIONS
;                                 *AC 3+1,*AC 3 + 1*.
;                                 RETURNS TO ACC 3 + 2.
; CALLS.                          NONE.
; CALLED-BY.                      EMS31,CVT54,PCH37,PI503.
; GLOBAL-VARIABLES-USED.          NONE.
; GLOBAL-VARIABLES-CHANGED.       NONE.
; ERROR-BITS-SET.                 NONE.

; SPACE FOR SAVING ACC CONTENTS.
AC047:    .BLK    1
AC147:    .BLK    1
AC247:    .BLK    1
AC347:    .BLK    1

MSK47:            17

; CHARACTER CODES FOR HEX CHARACTERS.
CHR47:            60
                  61
                  62
                  63
                  64
                  65
                  66
                  67
                  70
                  71
                  101
                  102
                  103
                  104
                  105
                  106
CHA47:            CHR47

; SAVE ACC CONTENTS,ENTRY.
```

```
;
CH147:  STA     0,AC047
        STA     1,AC147
        STA     2,AC247
        STA     3,AC347
;
;       AC0 C  CONTAINS NO TO BE  BE CONVERTED.
;       CONVERT NOBER BY SHIFTING ,MASKING,AND INDEXING.
;
;
;
        LDA     1,MSK47
        ADC     0,1
        LDA     2,CHA47
        ADD     1,2
        LDA     1,0,2
;
        MOVZR   0,0
        MOVZR   0,0
        MOVZR   0,0
        MOVZR   0,0
        LDA     3,MSK47
        ANDC    0,3
        LDA     2,CHA47
        ADD     3,2
        LDA     3,0,2
        MOVS    3,3
        ADC     1,3
        LDA     2,AC347
        STA     3,1,2
;;
        MOVZR   0,0
        MOVZR   0,0
        MOVZR   0,0
        MOVZR   0,0
        LDA     1,MSK47
        ANDC    0,1
        LDA     2,CHA47
        ADD     1,2
        LDA     1,0,2
;
        MOVZR   0,0
        MOVZR   0,0
        MOVZR   0,0
```

```
                LDZA    0,0
                LDA     3,MSK47
                LDA     2,3
                LDA     3,CHA47
                LDA     3,2
                LDA     0,0,2
                LDZS    0,0
                ADD     1,0
                LDA     3,AC347
                STA     0,0,3

;
; RESTORE ACC AND RETURN.
;
                LDA     0,AC047
                LDA     1,AC147
                LDA     2,AC247
                LDA     3,AC347
;
                JMP     2,3

;
;*****************************************************
; NAME.                         ESC32.
; FUNCTION.                     ENTER SOURCE CODE
;                               INTO PRTLN FROM 23-136.
; INPUT.                        NONE.
; OUTPUT.                       NONE.
; CALLS.                        PSC16.
; CALLED-BY.                    EPT33.
; GLOBAL-VARIABLES-USED.        INSLN.
; GLOBAL-VARIABLES-CHANGED.     PRPTR.
; ERROR-BITS-SET.               NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
AC032:  .BLK    1
AC132:  .BLK    1
```

```
AC232:  .BLK    1
AC332:  .BLK    1
;
; CONSTANTS AND COUNTERS.
;
CN.C32:         5.                      ; DEC 46
CN132:          33                      ; DEC 27
MSK32:          377
CTR32:  .BLK    1
;
; SAVE ACC CONTENTS,ENTRY.
;
ES132:  STA     0,AC032
        STA     1,AC132
        STA     2,AC232
        STA     3,AC332
;
; SET UP COUNTER TO COUNT UPTO 46 WORDS.
;
        LDA     0,CN032
        STA     0,CTR32
;
; SET  PRPTR VALUE.
;
        LDA     0,CN132
        STA     0,PRPTR
;
        LDA     2,INSLN
;
; GET NEXT WORD AND INTRODUCE IN PRTLN.
;
LOP32:  LDA     0,0,2
        MOVS    0,0
        LDA     1,MSK32
        AND     1,0
        JSR@    EPT+3                   ; LEFT HALF
;
;
        ISZ     PRPTR
        LDA     0,0,2
        LDA     1,MSK32
        AND     1,0
        JSR@    EPT+3                   ; RIGHT HALF.
;
; UPDATE COUNTERS,INDICES.
; CHECK WHETHER ALL WDS HAVE BEEN ENTERED.
;
```

```
          DSZ     PRPTR
          INC     2,2
          DSZ     CTR32
          JMP     LOP32

;RESTORE ACC CONTENTS AND RETURN.

          LDA     0,AC032
          LDA     1,AC132
          LDA     2,AC232
          JMPS    AC332

;*****************************************************

;NAME.                              EPT=3.
;FUNCTION.                          ENTERS THE CH. IN ACC 0
                                    INTO PRTLN,IN THE POSTION
                                    POINTED AT BY PRPTR.
;INPUT.                             CH. IN ACC 0.
;OUTPUT.                            NONE.
;CALLS.                             NONE.
;CALLED-BY.                         LOTS.
;GLOBAL-VARIABLES-USED.             PRPTR,PRTLN.
;GLOBAL-VARIABLES-CHANGED.          PRT_1.
;ERROR-BITS-SET.                    NONE.

;SPACE FOR SAVING ACC CONTENTS.

AC032:    .BLK    1
AC132:    .BLK    1
AC232:    .BLK    1
AC332:    .BLK    1

CNT32:              1                       ;DEC 1
MSK32:              377
```

```
MS183:          177400
;
; SAVE ACC CONTENTS,ENTRY.
;
EP183:  STA     0,AC083
        STA     1,AC183
        STA     2,AC283
        STA     3,AC383
;
; GET THE WORD POINTED TO BY PRPTR.
;
        LDA     0,PRPTR
        LDA     1,ON083                 ; DEC 1
        SUB     1,0
        MOVZR   0,0,SZC
;
; ACC 0 CONTAINS WD TO BE LOADED.
; VALUE BETWEEN 0 AND 59.
;
        JMP     NZ083                   ; RIGHT HALF
;
; CARRY IS ZERO,LEFT HALF TO BE REPLACED.
; ACC 0 CONTAINS THE WD TO BE LOADED.
;
        LDA     3,PRTLN
        ADD     0,3
        LDA     0,0,3
        LDA     1,MSK83                 ; LOAD MASK.
        AND     1,0                     ; LEFT HALF.
;
; AC 0 CONTAINS RIGHT HALF,LEFT HALF IS 0.
;
        LDA     1,AC183
        LDA     2,MSK83
        AND     2,1
        MOVS    1,1
;
; AC 1 CONTAINS RECD LEFT HALF,RIGHT HALF IS 0.
;
; FORM WD.
;
        ADD     1,0
        STA     0,0,3
        JMP     END83
;
; CARRY IS NON-ZERO,RIGHT HALF TO BE REPLACED.
; AC 0 CONTAINS WD TO BE REPLACED.
```

```
;
NZC83:  LDA     3,PRTLN
        ADD     0,3
        LDA     0,0,3
        LDA     1,MS183
        AND     1,0
;
; AC 0 CONTAINS LEFT HALF OF WD ,RIGHT HALF IS 0.
;
        LDA     1,AC083
        LDA     2,MSK83
        AND     2,1
;
; AC 1 CONTAINS RIGHT HALF,LEFT HALF IS 0.
; FORM WORD AND REPLACE.
;
        ADD     1,0
        STA     0,0,3
;
; RESTORE ACC CONTENTS AND RETURN.
;
END83:  LDA     0,AC083
        LDA     1,AC183
        LDA     2,AC283
        JMPS    AC383
;
;*****************************************************
;
; NAME.                         EPL20.
; FUNCTION.                     ENTER ERROR LETTERS UPTO A
;                               MAXIMUM OF 4 INTO
;                               PRTLN FROM 1-4.
; INPUT.                        NONE.
; OUTPUT.                       NONE.
; CALLS.                        EPT83.
; CALLED-BY.                    PSC10.
; GLOBAL-VARIABLES-USED.        PRTLN,PRERR.
; GLOBAL-VARIABLES-CHANGED.
```

```
;                                       PAPTR,PRT_1.
; ERROR-BITS-SET.                       NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC028:    .BLK    1
AC128:    .BLK    1
AC228:    .BLK    1
AC328:    .BLK    1
;
; CONSTANTS.
;
CN028:            20                          ; DEC 16
CN128:            1
CN228:            4
CTR28:    .BLK    1
CNT28:    .BLK    1
;
; CHARACTER CODES FOR ERROR LETTERS.
;
CHC28:            111
                  115
                  104
                  114
                  117
                  125
                  123
                  105
                  126
                  116
                  106
                  122
                  120
                  121
                  105
                  106
CHA28:            CHC28
;
; SAVE ACC CONTENTS,ENTRY
;
ER128:    STA     0,AC028
          STA     1,AC128
          STA     2,AC228
          STA     3,AC328
;
```

```
;
; SET UP COUNTERS TO COUNT UPTO 16,4
;
          LDA     3,CN328
          STA     3,CTR28
          LDA     3,CN228
          STA     3,CNT28
;
; SET PRPTR VALUE.
;
          LDA     3,CN128
          STA     3,PRPTR
;
; CHECK AND INTRODUCE INTO PRTLN.
;
          LDA     1,PRERR
          LDA     2,CHA28
;
LOP28:    MOVL    1,1,SNC
          JMP     ZCY28
;
; CARRY IS NON-ZERO,ENTER LETTER,UPDATE CTR.
;
          LDA     3,CTR28
          LDA     0,CN128
          SUB     0,3
          ADDC    2,3
          LDA     0,0,3
          JSR@    EPT83
          ISZ     PRPTR
          DSZ     CNT28
          JMP     .+2
          JMP     END28
;
; UPDATE COUNTER TO CNT UPTO 16.
;
ZCY28:    DSZ     CTR28
          JMP     LOP28
;
; RESTORE ACC CONTENTS AND RETURN.
;
END28:    LDA     0,ACC28
          LDA     1,AC128
          LDA     2,AC228
          JMP@    AC328
```

```
;
;*****************************************************************
;
; NAME.                       LOC30.
; FUNCTION.                   ENTERS LOCTN INTO PPTLN
;                             INTO POSITIONS 12-15 AFTER
;                             CONVERSION INTO HEX.
; INPUT.                      NONE.
; OUTPUT.                     NONE.
; CALLS.                      CVT64.
; CALLED-BY.                  PSC10.
; GLOBAL-VARIABLES-USED.      PPTLN,LCFLG,LOCTN.
; GLOBAL-VARIABLES-CHANGED.   PPTL1,PRPTR.
; ERROR-BITS-SET.             NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
AC030:   .BLK    1
AC130:   .BLK    1
AC230:   .BLK    1
AC330:   .BLK    1
;
; CONSTANTS.
CN030:           14                      ; DEC 12
;
; SAVE ACC CONTENTS,ENTRY.
LOC30:   STA     0,AC030
         STA     1,AC130
         STA     2,AC230
         STA     3,AC330
;
; SET PRPTR VALUE .
; SET ACC 0 VALUE TO LOCTN.
```

```
;CHECK WHETHER THE ENTERING IS NEED.

        LDA     0,CNT30
        STA     0,PRPTR
        LDA     0,LOCTN
        LDA     1,LCFLG
        MOV     1,1,SZR

;CALL ON ROUTINE TO CONVERT AND ENTER.

        JSRS    CVT64

;RESTORE ACC CONTENTS AND RETURN

        LDA     0,AC030
        LDA     1,AC130
        LDA     2,AC230
        JMPS    AC330

;*****************************************************

;NAME.                          CVT64.
;FUNCTION.                      CONVERT WORD IN ACC 3
;                               INTO 4 HEX CH. AND ENTER
;                               THEM INTO PATTN STARTING
;                               FROM PRPTR.
;INPUT.                         ACC 3.
;OUTPUT.                        NONE.
;CALLS.                         EPT33,SHX47.
;CALLED-BY.                     LOC33,EWS51.
;GLOBAL-VARIABLES-USED.         NONE.
;GLOBAL-VARIABLES-CHANGED.      PRPTR.
;ERROR-BITS-SET.                NONE.

;SPACE FOR SAVING ACC CONTENTS.
```

```
AC0364:    .BLK     1
AC1364:    .BLK     1
AC2364:    .BLK     1
AC3364:    .BLK     1
;
;
MSK364:              377
;
; SAVE ACC CONTENTS,ENTRY.
;
CV1364:    STA      0,AC0364
           STA      1,AC1364
           STA      2,AC2364
           STA      3,AC3364
;
; PRPTR VALUE ALREADY SET.
; ACC 0 CONTAINS WORD TO BE ENTERED.
;
;
; CONVERT INTO HEX,ACC 0 CONTAINS WORD.
;
           JSRS     BMX47
AG0364:    .BLK     1
AG1364:    .BLK     1
;
; ENTER WORD INTO PRTLN,CALL ON EPT63
;
           LDA      0,AG0364
           MOVS     0,0
           LDA      1,MSK364
           AND      1,0
           JSRS     EPT63
           ISZ      PRPTR
           LDA      0,AG0364
           LDA      1,MSK364
           AND      1,0
           JSRS     EPT63
           ISZ      PRPTR
           LDA      0,AG1364
           MOVS     0,0
           LDA      1,MSK364
           AND      1,0
           JSRS     EPT63
           ISZ      PRPTR
           LDA      0,AG1364
           LDA      1,MSK364
           AND      1,0
```

```
        JSRS    EPT83
;
; RESTORE ACC CONTENTS AND RETURN.
;
        LDA     0,AC086
        LDA     1,AC186
        LDA     2,AC286
        JMPS    AC386
;
;*****************************************************
;
; NAME.                         WRT86.
; FUNCTION.                     OUTS OUT A STRING OF
;                               CH. ,TERMINATED BY A NULL,
;                               PACKED RIGHT TO LEFT.
; INPUT.                         IN ACC 2,STARTING ADD.
; OUTPUT.                       NONE.
; CALLS.                        PUT7S.
; CALLED-BY.                    LOTS.
; GLOBAL-VARIABLES-USED.        NONE.
; GLOBAL-VARIABLES-CHANGED.     NONE
; ERROR-BITS-SET.               NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC086:  .BLK    1
AC186:  .BLK    1
AC286:  .BLK    1
AC386:  .BLK    1
;
MSK86:          177
;
; SAVE ACC CONTENTS,ENTRY POINT.
;
WR1C6:  STA     0,AC086
```

```
          STA     1,AC1B6
          STA     2,AC2B6
          STA     3,AC3B6
;
; GET NEXT WORD.AC2 CONTAINS ADDRESS.
;
LCPB6:    LDA     1,0,2
          LDA     0,MSKB6
          AND     1,0,SNR
          JMP     ENDB6                   ;CH. IS 0
          JSRS    PUT7B
          MOVS    1,1
          LDA     0,MSKB6
          AND     1,0,SNR
          JMP     ENDB6                   ; CH. IS 0.
          JSRS    PUT7B
          INC     2,2
          JMP     LCPB6
;
; ALL CHARACTERS PRINTED OUT,RETURN.
;
ENDB6:    LDA     0,AC0B6
          LDA     1,AC1B6
          LDA     2,AC2B6
          JMPS    AC3B6
;
;*****************************************************
;
; NAME.                         MODB6.
; FUNCTION.                     PRINTS OUT MODE= QUERY.
;                               GETS REPLY AND SETS BNFLG,
;                               LTFLG,ENDAD ACCORDINGLY.
; INPUT.                        NONE.
; OUTPUT.                       NONE.
; CALLS.                        PUT6,REC7A.
; CALLED-BY.                    ASSMB
; GLOBAL-VARIABLES-USED.        ECOTO,MSING.
; GLOBAL-VARIABLES-CHANGED.     INPDV,OUTDV,ENDAD,
```

```
;                                      LTFLG, NEFLG.
; ERROR-BITS-SET.
;                                      NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC085:     .BLK      1
AC185:     .BLK      1
AC285:     .BLK      1
AC385:     .BLK      1
;
MES85:     .TXT      .<015><012> MODE := .
MEA85:               MES85
;
CN085:               0
CN185:               1
CN285:               60
CRT85:               15
LFC85:               12
;
; SAVE ACC CONTENTS ,ENTRY.
;
MO185:     STA       0,AC085
           STA       1,AC185
           STA       2,AC285
           STA       3,AC385
           LDA       0,MOIND
           STA       0,INPOV
           LDA       0,MOOTC
           STA       0,OUTOV
;
; PRINT OUT QUERY.
;
RPT85:     LDA       2,MEA85
           JSR$      WRT66
;
; GET REPLY.
;
           JSR$      RED74
;
; REPLY IS IN ACC 0.
; CHECK WHAT REPLY IS AND TAKE ACTION.
;
           LDA       1,CN285                    ; 60 OCTAL
           SUB       1,0,SNR
           HALT
;
```

```
        LDA     1,CN185
        SUB     1,0,SNR
        JMP     MON85                   ; MODE IS 1
        SUB     1,0,SNR
        JMP     MTN85                   ; MODE IS 2
        SUB     1,0,SNR
        JMP     MTH85                   ; MODE IS 3
        JMP     RPT85                   ; REPEAT QUERY.
;
; MODE = 1,SET ENDAD TO 0,RETURN.
;
MON85:  LDA     0,CN085
        STA     0,ENDAD
        JMP     END85
;
; MODE = 2,SET ENDAD TO 1,LTFLG TO 0,BNFLG TO 1.
;
MTN85:  LDA     0,CN085
        STA     0,LTFLG
        LDA     0,CN185
        STA     0,ENDAD
        STA     0,BNFLG
        JMP     END85
;
; MODE IS 3,SET ENDAD TO 1,LTFLG TO 1,BNFLG TO 0.
;
MTH85:  LDA     0,CN085
        STA     0,BNFLG
        LDA     0,CN185
        STA     0,ENDAD
        STA     0,LTFLG
;
; PRINT OUT LF AND CR.
; RESTORE ACC CONTENTS AND RETURN.
;
END85:  LDA     0,CRT85
        JSR@    PUT75
        LDA     0,LFD85
        JSR@    PUT75
        LDA     0,AC085
        LDA     1,AC185
        LDA     2,AC285
        JMP@    AC385
;
;
;*****************************************************************
;
```

```
;NAME.                          END52.
;FUNCTION.                      SERVICE THE PSEUDO-OP
                                NON,END.SET ENDFG,PNFLG.
;INPUT.                         NONE.
;OUTPUT.                        NONE.
;CALLS.                         END26.
;CALLED-BY.                     SA 25.
;GLOBAL-VARIABLES-USED.         LBFLG.
;GLOBAL-VARIABLES-CHANGED.      ENDFG,LOCFLG,PNFLG.
;ERROR-BITS-SET.                BIT 10 12(P).

; SPACE FOR SAVING ACC CONTENTS.

AC052:    .BLK    1
AC152:    .BLK    1
AC252:    .BLK    1
AC352:    .BLK    1
;
TWL52:            14                    ; DEC 12
;
; SAVE ACC CONTENTS,ENTRY.
;
EN152:    STA     0,AC052
          STA     1,AC152
          STA     2,AC252
          STA     3,AC352
;
; SET LOCFLG TO 0.
; SET ENDAD,PNFLG TO 1.
;
          SUB     0,0
          STA     0,LOCFLG
          INC     0,0
          STA     0,ENDFG
          STA     0,PNFLG
;
; CHECK LBFLG TO SEE IF ERROR.
```

```
;
        LDA     0,LBFLG
        MOV     0,0,SNR
        JMP     STP52
;
; SET BIT NO. 12(P).
;
        LDA     0,TWL52
        STA     0,ARG52
        JSR@    ERE26
ARG52:  .BLK    1
;
; RESTORE ACC CONTENTS AND RETURN.
;
STP52:  LDA     0,AC052
        LDA     1,AC152
        LDA     2,AC252
        JMP@    AC352
;
;
;*******************************************************
;
; NAME.                             ABA71.
; FUNCTION.                         SERVICES INHERENT TYPE
;                                   OF ADDRESSING.FORMAT NO.6.
; INPUT.                            NONE.
; OUTPUT.                           NONE.
; CALLS.                            NONE.
; CALLED-BY.                        SPM25.
; GLOBAL-VARIABLES-USED.            BSVAL.
; GLOBAL-VARIABLES-CHANGED.         LENTH,WSPCE.
; ERROR-BITS-SET.                   NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC071:  .BLK    1
;
```

```
CN171:          1
;
; SAVE AC0 CONTENTS AND ENTRY POINT.
;
AB171:  STA     0,AC071
;
; SET LENTH TO 1
;
        LDA     0,CN171
        STA     0,LENTH
;
; SET WSPCE.
;
        LDA     0,BSVAL
        MOVS    0,0
        STAS    0,WSPCE
;
; RESTORE AC0 CONTENTS,RETURN.
;
        LDA     0,AC071
        JMP     0,3
;
;****************************************************************
;
; NAME.                             BIN15.
; FUNCTION.                         TAKES CARE OF OBJECT-CODE.
;                                   FILLS UP BUFFER AND FLUSHES
;                                   IT PERIODICALLY OR IF
;                                   PNFLG IS SET.
; INPUT.                            NONE.
; OUTPUT.                           NONE.
; CALLS.                            IBN35,EBN36,PCN37.
; CALLED-BY.                        PASS2,FC055,FD056,FC354.
; GLOBAL-VARIABLES-USED.            BNFLG,PNFLG,NOWDS.
; GLOBAL-VARIABLES-CHANGED.         LOCAL.
; ERROR-BITS-SET.                   NONE.
;
```

```
;
;  SPACE FOR SAVING ACC CONTENTS.
;
AC015:    .BLK      1
AC115:    .BLK      1
AC215:    .BLK      1
AC315:    .BLK      1
;
;
CN115:              1
;
;  ENTRY POINT,SAVE ACC CONTENTS.
;
EI115:    STA       0,AC015
          STA       1,AC115
          STA       2,AC215
          STA       3,AC315
;
;  CHECK BNFLG .
;
          LDA       0,BNFLG
          LDA       1,CN115
          SUBBE     0,1,SZR
          JMP       END15
;
;  CHECK IF LOCAL IS 0.
;
          LDA       0,LOCAL
          MOV       0,0,SNR
          JSRS      IEN35
;
;  CHECK PNFLG.
;
          LDA       0,PNFLG
          LDA       1,CN115
          SUBBE     0,1,SNR
          JMP       PCH15
;
;  CALL ON ROUTINE TO ENTER NORCE INTO BNOUT.
;  CHECK LOCAL AGAINST NCWDS.
;
          JSRS      EEN36
          LDA       0,NCWDS
          LDA       1,LOCAL
          SUBBLE    0,1,SNC
PCH15:    JSRS      PCH37
;
```

```
; RESTORE ACC CONTENTS AND RETURN.
;
END15:    LDA     0,AC015
          LDA     1,AC115
          LDA     2,AC215
          JMP$    AC315
;
;
;*****************************************************************
;
; NAME.
;                                    ISN35.
; FUNCTION.
;                                    INITIALISE BNOUT,(OBJECT-
;                                    -CODE BUFFER).
;                                    CALLED WHEN LOCAL IS 0.
;                                    CLEARS BNOUT,SETS S1,SETS
;                                    ADDRESS AND SETS LOCAL TO 0
; INPUT.
;                                    NONE.
; OUTPUT.
;                                    NONE.
; CALLS.
;                                    NONE.
; CALLED-BY.
;                                    BIN15.
; GLOBAL-VARIABLES-USED.
;                                    BNOUT,LOCTR.
; GLOBAL-VARIABLES-CHANGED.
;                                    LOCAL,BNOU1.
; ERROR-BITS-SET.
;                                    NONE.
;
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC035:    .BLK    1
AC135:    .BLK    1
AC235:    .BLK    1
AC335:    .BLK    1
;
; CONSTANTS.
;
CN035:            0
CN135:            51461
CN335:            50
CN435:            4
```

```
CTR35:  .BLK    1
MSK35:          377
;
; SAVE ACC CONTENTS,ENTRY.
;
IB135:  STA     0,AC035
        STA     1,AC135
        STA     2,AC235
        STA     3,AC335
;
; INITIALISE BNOUT.
;
        LDA     0,ON335
        STA     0,CTR35
;
        LDA     2,BNOUT
        LDA     0,ON035
;
LB135:  STA     0,0,2
        INC     2,2
        DSZ     CTR35
        JMP     LB135
;
; SET S1.
;
        LDA     0,ON135
        LDA     2,BNOUT
        STA     0,0,2
;
; SET ADDRESS VALUE.
;
        LDA     0,LOCTR
        MOVS    0,0
        LDA     1,MSK35
        ANC     1,0
        STA     0,2,2
        LDA     0,LOCTR
        ANC     1,0
        STA     0,3,2
;
; SET LOCAL TO 4
;
        LDA     0,ON435
        STA     0,LOCAL
;
; RESTORE ACCS AND RETURN.
;
```

```
        LDA     0,AC036
        LDA     1,AC136
        LDA     2,AC236
        JMPS    AC336

;*****************************************************

; NAME.                                 EB136.
; FUNCTION.                             ENTERS LENTH BYTES FROM
;                                       WORDS INTO BNOUT.
;                                       UPDATES LOCAL.
; INPUT.                                NONE.
; OUTPUT.                               NONE.
; CALLS.                                NONE.
; CALLED-BY.                            BIN15.
; GLOBAL-VARIABLES-USED.                BNOUT,LENTH.
; GLOBAL-VARIABLES-CHANGED.             LOCAL,BNOU1.
; ERROR-BITS-SET.                       NONE.

; SPACE FOR SAVING ACC CONTENTS.

AC036:  .BLK    1
AC136:  .BLK    1
AC236:  .BLK    1
AC336:  .BLK    1
CTR36:  .BLK    1
MSK36:          377

; SAVE ACC CONTENTS,ENTRY.

EB136:  STA     0,AC036
        STA     1,AC136
        STA     2,AC236
        STA     3,AC336

; ENTER LENTH BYTES INTO BNOUT.
```

```
;
            LDA     0,LENTH
            STA     0,CTR36
            NEG     0,0,SNR
            JMP     END36
            LDA     3,BNOUT
            LDA     0,LOCAL
            ADD     0,3
;
            LDA     2,WSPCE
            LDA     0,0,2
            MOVS    0,0
            LDA     1,MSK36
            AND     1,0
            STA     0,0,3
            DSZ     CTR36
            JMP     .+2
            JMP     END36
;
;
            LDA     0,,,2
            AND     1,0
            STA     0,1,3
            DSZ     CTR36
            JMP     .+2
            JMP     END36
;
;
            LDA     0,1,2
            MOVS    0,0
            AND     1,0
            STA     0,2,3
;
;  SET LOCAL RESTORE ACC CONTENTS,RETURN.
;
END36:      LDA     0,LOCAL
            LDA     1,LENTH
            ADD     1,0
            STA     0,LOCAL
;
            LDA     0,AC036
            LDA     1,AC136
            LDA     2,AC236
            JMPS    AC336
;
;
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

```
;NAME.                          SUB50.
;FUNCTION.                      GETS CHARACTER,ENTERS IT
;                               INTO BINOUT AND UPDATES
;                               LOCAL.
;INPUT.                         NONE.
;OUTPUT.                        NONE.
;CALLS.                         NONE.
;CALLED-BY.                     PDH37.
;GLOBAL-VARIABLES-USED.         BINOUT.
;GLOBAL-VARIABLES-CHANGED.      BLOCK,LOCAL.
;ERROR-BITS-SET.                NONE.

; SPACE FOR SAVING ACC CONTENTS.

AC050:    .BLK    1
AC150:    .BLK    1
AC250:    .BLK    1
AC350:    .BLK    1
CTR50:    .BLK    1
MSK50:            377

; SAVE ACC CONTENTS,ENTRY

SU150:    STA     0,AC050
          STA     1,AC150
          STA     2,AC250
          STA     3,AC350
;
          LDA     0,LOCAL
          STA     0,CTR50
          INC     0,0,SNR
          JMP     END50
;
          SUBZ    0,0
          DSZ     CTR50
          JMP     .+2
```

```
          JMP     ENDSU
;
          LDA     2,3NOUT
          INC     2,2
          LDA     0,1,2
          DSZ     CTR50
          JMP     .+2
          JMP     ENS50
;
LBL50:    INC     2,2
          LDA     1,0,2
          ADD     1,0
          DSZ     CTR50
          JMP     LBL50
;
;  ACC 0 CONTAINS THE SUM.MASK OUT AND COMPLIMENT FOR CHECKSUM.
;
ENS50:    COM     0,0
          LDA     1,MSK50
          AND     1,0
          STA     0,1,2
          ISZ     LOCAL
;
;  RESTORE ACC CONTENTS AND RETURN.
;
          LDA     0,AC050
          LDA     1,AC150
          LDA     2,AC250
          JMPS    AC350
;
;*****************************************************************
;
;  NAME.                        PCH37.
;
;  FUNCTION.                    FLUSHES THE OBJECT CODE
;                               BUFFER,3NOUT.
;                               GETS CHECKSUM ,CONVERTS
;                               INTO HEX. AND PUNCHES OUT.
;
;  INPUT.                       NONE.
;
;  OUTPUT.                      NONE.
;
;  CALLS.                       SUM50,PUN40,BHX47.
```

```
;  CALLED-BY.                          BIN15.
;  GLOBAL-VARIABLES-USED.              BNOUT.
;  GLOBAL-VARIABLES-CHANGED.           LOCAL,BNOL1.
;  ERROR-BITS-SET.                     NONE.
;
;  SPACE FOR SAVING ACC CONTENTS.
;
AC037:    .BLK     1
AC137:    .BLK     1
AC237:    .BLK     1
AC337:    .BLK     1
;
;  CONSTANTS
;
CN0237:             0
CN2237:             2
CTE37:    .BLK     1
FIV37:              5
;
;  STORE ACC CONTENTS,ENTRY.
;
PC137:    STA      0,AC037
          STA      1,AC137
          STA      2,AC237
          STA      3,AC337
;
;  GET CHECKSUM
;
;  INTRODUCE BYTE COUNT.
;
          LDA      2,BNOUT
          LDA      0,LOCAL
          INC      0,0
          LDA      1,CN2237
          SUB      1,0
          STA      0,1,2
          JSR@     SC15J
;
;  CONVERT BINOUT
;
          LDA      0,LOCAL
```

```
            LDA     1,FI/37
            SUBE    0,1,SNR
            JMP     NXT37
            STA     0,CTR37
            LDA     2,BNOUT
            DSZ     CTR37
            INC     2,2
;
LBL37:      LDA     0,0,2
            JSR$    BHX47
AG137:      .BLK     1
AG237:      .BLK     1
            LDA     0,AG237
            STA     0,0,2
            INC     2,2
            DSZ     CTR37
            JMP     LBL37
;
; ALL WDS ARE CONVERTED,PUNCH OUT
;
            JSR$    PNL12
            JSR $   PUN40
            JSR$    PNL12
;
; SET LOCAL TO 0.
;
NXT37:      LDA     0,CN037
            STA     0,LOCAL
;
;RESTORE ACC CONTENTS AND RETURN.
;
            LDA     0,AC037
            LDA     1,AC137
            LDA     2,AC237
            JMP$    AC337
;
;
;***************************************************************
;
; NAME.                         PUN40.
;
; FUNCTION.                     PUNCHES OUT ALL WORDS
;                               IN BNOUT FROM 1 TO LOCAL.
;
; INPUT.                        NONE.
;
; OUTPUT.
```

```
;  CALLS.                          NONE.
;  CALLED-BY.                      PCH37.
;  GLOBAL-VARIABLES-USED.          PUT75.
;  GLOBAL-VARIABLES-CHANGED.       SPRDV,LOCAL,SHOUT.
;  ERROR-BITS-SET.                 OUTDV.
;                                  NONE.

;  SPACE FOR SAVING ACC CONTENTS.

AC040:    .BLK     1
AC140:    .BLK     1
AC240:    .BLK     1
AC340:    .BLK     1

;  CONSTANTS.

CTR40:    .BLK     1
MSK40:             377

;  SAVE ACC CONTENTS.

PU140:    STA      0,AC040
          STA      1,AC140
          STA      2,AC240
          STA      3,AC340

;
;  PUNCH OUT NULLS.

          JSRS     PNL12

;  SET DEVICE CODE.

          LDA      0,SPRDV
          STA      0,OUTDV

;  SET UP COUNTER.

          LDA      0,LOCAL
          STA      0,CTR40
;
```

```
        LDA     2,BNOUT
;
; PUNCH OUT.
;
        LDA     1,MSK40
LBL40:  LDA     0,0,2
        MOVS    0,0
        ANDC    1,0
        JSR@    PUT75
        LDA     0,0,2
        ANDC    1,0
        JSR@    PUT75
        INC     2,2
        DSZ     CTR40
        JMP     LBL40
;
; PUNCH OUT NULLS.
;
        JSR@    PNL12
;
; RESTORE ACC CONTENTS AND RETURN.
;
        LDA     0,AC040
        LDA     1,AC140
        LDA     2,AC240
        JMP@    AC340
;
;*****************************************************
;
; NAME.                         GCH67
; FUNCTION.
;                               GET NEXT CH. FROM
;                               INSLN.
; INPUT.
;                               CHPTR POINTING TO CH. BEFOR
;                               REQD. ONE.
; OUTPUT.
;                               RETURNS THE CH. IN ACC 0.
;                               RETURNS A BLANK IF GREATER
;                               THAN 60.
; CALLS.
;                               GCHS1 .
; CALLED-BY.
;                               LDTS.
; GLOBAL-VARIABLES-USED.
```

```
;
;  GLOBAL-VARIABLES-CHANGED.  INCHR.
;                             CHPTR.
;  ERROR-BITS-SET.            NONE.
;
;
;  SPACE FOR SAVING ACC CONTENTS.
;
AC187:    .BLK    1
AC287:    .BLK    1
AC387:    .BLK    1
;
;  CONSTANTS.
;
CNS87:            121                     ; DEC 81
BLK87:            40                      ; BLANK
;
;  ENTRY POINT,SAVE ACCS.
;
GN187:    STA     2,AC287
          STA     1,AC187
          STA     3,AC387
;
;  INC CHPTR AND CHECK.
;
          ISZ     CHPTR
          LDA     0,CHPTR
          LDA     3,CNS87
          SUBLE   3,0,SNC
          JMP     ERR87
;
;  CHPTR OKAY,GET CH.
;
          JSRS    GCH81
;
;  AC 0 CONTAINS CH.RETURN.
;
          JMP     END87
;
;  CHPTR IS 81.
;
ERR87:    LDA     0,BLK87
          STA     3,CHPTR
;
;  RESTORE ACC .RETURN.
;
```

```
ENC87:   LDA     1,AC187
         LDA     2,AC287
         JMPS    AC387
;
;
;
;*****************************************************
;
; NAME.
;                                 BNY93.
; FUNCTION.
;                                 CONVERTS A SERIES OF ASCII
;                                 BIN. CH. INTO A BINARY NO.
; INPUT.
;                                 CHPTR POINTS TO CH.
;                                 BEFORE STRING.
;                                 CH. ARE REQUESTED THRO
;                                 A CALL TO GNC87.
; OUTPUT.
;                                 CHPTR POINTS AT BREAK.
;                                 ACC 0 CONTAINS BREAK.
;                                 ACC 1 CONTAINS VALUE.
; CALLS.
;                                 GNC87.
; CALLED-BY.
;                                 CVT96.
; GLOBAL-VARIABLES-USED.
;                                 NONE.
; GLOBAL-VARIABLES-CHANGED.
;                                 NONE.
; ERROR-BITS-SET.
;                                 NONE.
;
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC293:   .BLK    1
AC393:   .BLK    1
;
; CONSTANTS.
;
CH093:           61                          ;CH 1
CH793:           60                          ; CH 0
FSM93:   .BLK    1
;
; SAVE ACC,ENTRY,INITIALISE RUNNING SUM.
;
BN193:   STA     2,AC293
```

```
        STA     3,AC393
        SUB     0,0
        STA     0,RSM93
;
; GET NEXT CH AND UPDATE RUNNING SUM.
;
LBL93:  JSRS    GNC87
        LDA     1,CHZ93
        LDA     2,CHO93
        ADCZE   2,J,SNC
        ADCZE   0,1,SZC
        JMP     BRK93
;
        SUB     1,J                     ; REDUCE CH
        LDA     1,RSM93
;
; MULTIPLY BY 2 AND ADD
;
        MOVZL   1,1
        ADD     J,1
        STA     1,RSM93
;
        JMP     LBL93
;
; ACC 0 CONTAINS BREAK CH ,RETURN .
;
BRK93:  LDA     1,RSM93
        LDA     2,AC293
        JMPS    AC393
;
;*****************************************************
;
; NAME.                         HEX94.
; FUNCTION.                     CONVERTS A SERIES OF ASCII
;                               HEX CH. INTO A BINARY NO.
; INPUT.                        CHPTR POINTS TO CH.
;                               BEFORE STRING.
;                               CH. ARE REQUESTED THRO
;                               A CALL TO GNC87.
; OUTPUT.                       CHPTR POINTS AT BREAK.
;                               ACC 0 CONTAINS BREAK.
```

```
;                                       ACC 1 CONTAINS VALUE.
; CALLS.                               GNC87.
; CALLED-BY.                           CVT86.
; GLOBAL-VARIABLES-USED.               NONE.
; GLOBAL-VARIABLES-CHANGED.            NONE.
; ERROR-BITS-SET.                      NONE.

; SPACE FOR SAVING ACC CONTENTS.

AC294:   .BLK    1
AC394:   .BLK    1

; CONSTANTS.

CNS94:           67                     ;USED TO
                                        ;REDUCE A-F.
CHZ94:           60                     ; CH 0
CH994:           71                     ; CH 9
CHA94:           101                    ; CH A
CHF94:           106                    ; CH F
RSM94:   .BLK    1                      ; RUNNING SUM

; SAVE ACC,ENTRY,INITIALISE RSM94

HE194:   STA     2,AC294
         STA     3,AC394
         SUB     0,0
         STA     0,RSM94

; GET NEXT CHARACTER,CHECK IT OUT,MULTIPLY BY
; 16 AND ADD .

LBL94:   JSRS    GNC87
         LDA     1,CHZ94
         LDA     2,CH994
         ADCZE   2,0,SNC
         ADCZE   0,1,SZC
         JMP     CHK94
         JMP     MAD94

; CH NOT NUMERIC,CHECK IF IT IS IN A-F.
```

```
;
CHK94:  LDA     1,CHA94
        LDA     2,CHF94
        ADCCZE  2,J,SNC
        ADCZE   J,1,SZC
        JMP     BRK94
        LDA     1,CNS94                 ; VALUE USED TO
;                                       REDUCE A-F.
;
; CH IS OKAY,REDUCE ,MULTIPLY,AND ADD.
;
MAC94:  SUB     1,J
        LDA     1,RSV94
        MOVZL   1,1
        MOVZL   1,1
        MOVZL   1,1
        MOVZL   1,1
        ADD     0,1
        STA     1,RSV94
        JMP     LEL94
;
; AC 0 CONTAINS BREAK CH.RESTORE ACC,RETURN.
;
BRK94:  LDA     1,RSV94
        LDA     2,AC294
        JMPS    AC394
;
;*****************************************************
;
; NAME.                         OCT92.
; FUNCTION.                     CONVERTS A SERIES OF ASCII
;                               DECIMAL CH. INTO A BINARY
;                               NUMBER.
; INPUT.                        CHPTR POINTS TO CH.
;                               BEFORE STRING.
;                               CH. ARE REQUESTED THRO
;                               A CALL TO GNC87.
; OUTPUT.                       CHPTR POINTS AT BREAK.
;                               ACC 0 CONTAINS BREAK.
;                               ACC 1 CONTAINS VALUE.
; CALLS.                        GNC87.
```

```
;
; CALLED-BY.                          CVT96.
; GLOBAL-VARIABLES-USED.              NONE.
; GLOBAL-VARIABLES-CHANGED.           NONE.
; ERROR-BITS-SET.                     NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC292:    .BLK    1
AC392:    .BLK    1
;
; CONSTANTS
;
CHS92:            67                  ; CH 7
CH792:            60                  ;  CH
RSM92:    .BLK    1
;
; SAVE ACS ,ENTRY,INITIALISE RUNNING SUM.
;
CC192:    STA     2,AC292
          STA     3,AC392
          SUB     0,0
          STA     0,RSM92
;
; GET NEXT CH AND UPDATE RUNNING SUM.
;
LOL92:    JSR@    GNC97
          LDA     1,CHZ92
          LDA     2,CHS92
          ADCZE   2,0,SNC
          ADCZE   0,1,SZC
          JMP     BRK92
;
;
          SUB     1,0
          LDA     1,RSM92
;
; MULTIPLY BY 6 AND ADD
;
          MOVZL   1,1
          MOVZL   1,1
          MOVZL   1,1
          ADD     0,1
```

```
;
          STA     1,RSV92
          JMP     LBL92
;
; ACC 0 CONTAINS BREAK CH, SET ACC 1 VALUE AND RETURN.
;
BRK92:    LDA     1,RSV92
          LDA     2,AC292
          JMPS    AC392
;
;*****************************************************
;
; NAME.                          DEC91.
; FUNCTION.                      CONVERTS A SERIES OF ASCII
;                                OCTAL CH. INTO A BINARY
;                                NUMBER.
; INPUT.                         CHPTR POINTS TO CH.
;                                BEFORE STRING.
;                                CH. ARE REQUESTED THRO
;                                A CALL TO GNC97.
; OUTPUT.                        CHPTR POINTS AT BREAK.
;                                ACC 0 CONTAINS BREAK.
;                                ACC 1 CONTAINS VALUE.
; CALLS.                         GNC97.
; CALLED-BY.                     CVT96,PAG60,NER17.
; GLOBAL-VARIABLES-USED.         NONE.
; GLOBAL-VARIABLES-CHANGED.      NONE.
; ERROR-BITS-SET.                NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC291:    .BLK    1
AC391:    .BLK    1
;
; CONSTANTS.
;
```

```
CHN91:          71                      ; CH 9
CHZ91:          60                      ; CH 0
RSM91:  .BLK    1                       ; RUNNING SUM.
;
; SAVE ACC CONTENTS,ENTRY POINT,INITIALISE RS 91.
;
CE191:  STA     2,AC291
        STA     3,AC391
        SUB     0,0
        STA     0,RSM91
;
;
; EVALUATE BY SUCCESSIVE CALLS TO GNC87 AND MULTIPLYING
; BY 10.
;
LBL91:  JSRS    GNC87
        LDA     1,CHZ91
        LDA     2,CHN91
        ADCZE   2,0,SNC                 ; SKIP IF > 9
        ADCZE   0,1,SZC                 ; SKIP IF >= 0
        JMP     BRK91
;
; CH IS NOT BREAK
;
        SUB     1,0
        LDA     1,RSM91
;
; MULTIPLY BY 10 AND ADD
;
        MOVZL   1,2
        MOVZL   2,2
        ADC     2,1
        MOVZL   1,1
        ADD     0,1
;
;
        STA     1,RSM91
        JMP     LBL91
;
; ACC 0 CONTAINS BREAK CH.,RETURN.
;
BRK91:  LDA     1,RSM91
        LDA     2,AC291
        JMPS    AC391
;
```

```
;*****************************************************************
;
; NAME.
;                                SYM95.
; FUNCTION.
;                                GETS THE VALUE OF
;                                THE SYMBOL FROM SYMTB.
; INPUT.
;                                CHPTR POINTS TO CH.
;                                BEFORE SYMBOL.
; OUTPUT.
;                                CHPTR POINTS TO BREAK CH.
;                                ACC 0 CONTAINS BREAK (=0 IF
;                                ERROR).
;                                ACC 1 CONTAINS VALUE.
; CALLS.
;                                STB79,GNC87,ENE26.
; CALLED-BY.
;                                CVT96.
; GLOBAL-VARIABLES-USED.
;                                LOCTR,PSFLG.
; GLOBAL-VARIABLES-CHANGED.
;                                NONE.
; ERROR-BITS-SET.
;                                5(S),5(U).
;
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC295:   .BLK      1
AC395:   .BLK      1
;
; CONSTANTS
;
AST95:             52
CST95:             1
CN595:             5
CN695:             6
CNT95:             7
;
; COUNTERS AND SPACE FOR STORING LABEL
;
CTR95:   .BLK      1
LBL95:   .BLK      3
LBA95:             LBL95
;
; SAVE ACC CONTENTS,ENTRY
```

```
;
SY195:  STA     2,AC295
        STA     3,AC395
        JSRS    GNC87
        LDA     1,AST95
        SUBE    0,1,SNR
        JMP     ASK95
        DSZ     CHRTR
;
;;; SET UP COUNTER TO COUNT UPTO 6
;
        LDA     0,CNT95
        STA     0,CTR95
;
;;; CLEAR LABEL SPACE
;
        LDA     2,LBA95
        SUB     0,0
        STA     0,0,2
        STA     0,1,2
        STA     0,2,2
;
;;; GET CH AND SAVE
;
LOP95:  JSRS    GNC87
        JSRS    CL376
        JMP     BRK95
;
;;; CH IS ALPHANUMERIC,CHECK FOR LENGTH.
;
        DSZ     CTR95
        JMP     .+2
        JMP     ERR95
;
;;; CHECK WHICH HALF
;
        LDA     3,CT295
        MOVR    3,3,SNC
        JMP     ZCY95
;
;;; CARRY IS NON ZERO,STORE AND UPDATTE ACC 2
;
        ADD     1,0
        STA     0,0,2
        INC     2,2
        JMP     LOP95
```

```
;
;  CARRY IS ZERO,SAVE IN ACC 1
;
ZCY95:  MOVS    0,1
        JMP     LCP95
;
;  CH IN ACC 0 IS BREAK,SEARCH TABLE AFTER POSSIBLY
;  SAVING ACC 1
;
B-K95:  LDA     3,CT695
        STA     0,BREAK
        MOV#    3,3,SNC
        STA     1,0,2
;
        LDA     1,LBA95
        STA     1,AG095
        LDA     1,CST95
        STA     1,AG195
        JSR@    ST379
AG095:  .BLK    1
AG195:  .BLK    1
AG295:  .BLK    1
FLG95:  .BLK    1
;
;  CHECK WHETHER FOUND OR NOT
;
        LDA     1,AG295
        LDA     2,FLG95
        MOVLE   2,2,SNC
        JMP     END95
;
;  NOT FOUND.
;  ENTER ERROR NO 5
;  IF PASS IS 2.
;  SET NULL BREAK CHARACTER.
;
        LDA     2,PSFLG
        LDA     1,CST95
        SUBE    2,1,SNR
        JMP     SET95                   ; DON'T SET
                                        ERROR BIT.
;
;  PASS IS 2,SET ERROR BIT 5.
;
        LDA     2,ON595
        STA     2,AG395
        JSR@    ENE25
```

```
AG395:    .BLK    1
;
SET95:    SUB     0,0
          SUB     1,1
          JMP     END95
;
; SYMBOL TOO LONG,ENTER ERROR BIT 0.
;
ERR95:    LDA     2,ON595
          STA     2,AG495
          JSRS    ENE26
AG495:    .BLK    1
          SUB     1,1
          SUB     0,0
          JMP     END95
ASK95:    LDA     1,LOCTS
          JSRS    GNC07
;
; RESTORE ACC,RETURN
;
END95:    LDA     2,AC295
          JMPS    AC395
;
;*****************************************************
;
; NAME.                          EVL97.
; FUNCTION.                      EVALUATE AN EXPN.
; INPUT.                         CHPTR POINTS TO FIRST CH.
;                                OF EXPN.
; OUTPUT.                        CHPTR POINTS TO BREAK.
;                                ACC 0 CONTAINS BREAK,NULL
;                                IF ERROR.
;                                ACC 1 CONTAINS VALUE.
; CALLS.                         ENE26,CVT96,ACD93,
;                                SUB69,MPY72,DIV96.
; CALLED-BY.                     LOTS.
; GLOBAL-VARIABLES-USED.         NONE.
; GLOBAL-VARIABLES-CHANGED.
```

```
;
; ERROR-BITS-SET.                    CHPTR.
;                                    7(C).
;
;
; SPACE FOR SAVING ACC CONTENT.
;
AC297:    .BLK     1
AC397:    .BLK     1
; OTHER CONSTANTS.
PLS97:             53                    ; CH +
MIN97:             55                    ; CH -
SLH97:             57
AST97:             52
BLK97:             40                    ;CH. BLANK
COM97:             54                    ;CH. COMMA
SEV97:             7                     ;SEC  7
;
; SPACE FOR BREAK,VALUE
; AND ADDRESS OF SERVICE ROUTINE.
;
ADR97:    .BLK     1
BRK97:    .BLK     1
VAL97:    .BLK     1
;
; SAVE ACC CONTENTS,ENTRY,SET CHPTR.
;
EV197:    STA      2,AC297
          STA      3,AC397
          DSZ      CHPTR
;
; INITIALISE VAL97,BRK97 TO 0,+
;
          SUB      0,J
          STA      0,VAL97
          LDA      0,PLS97
          STA      0,BRK97
;
; CHECK BREAK CHARACTER .
;
LBL97:    LDA      0,BRK97
          MOVE     0,J,SZR
          STA      0,BREAK
;
; CHECK IF I IS +
;
          LDA      1,PLS97
```

```
          SUBE    0,1,SNR
          JMP     ADD97                   ; CHARACTER IS +
;
; CHECK IF IT IS -
;
          LDA     1,MIN97
          SUBE    0,1,SNR
          JMP     SUB97
;
; CHECK IF  IT IS  SLASH.
;
          LDA     1,SLH97
          SUBE    0,1,SNR
          JMP     DIV97                   ; CH. IS /.
;
; CHECK IF CH. IS *
;
          LDA     1,AST97
          SUBE    0,1,SNR
          JMP     MUL97                   ; CH IS *
;
; CHECK IF IT IS BLANK OR COMMA.
;
          LDA     1,BLK97
          SUBE    0,1,SNR
          JMP     NXT97
          LDA     1,COM97
          SUBE    0,1,SNR
          JMP     NXT97                   ; CH IS COMMA
;
; CHAR,ACTER IS NONE OF ABOVE,SEE IF IT IS
; A NULL. IF SO DO NOTT SET ERROR BIT SINCE
; IT HAS BEEN ALREADY SET.
;
          MOV     0,0,SNR
          JMP     ER197
;
; CHARACTER IS NONE OF ABOVE.SET ERROR BIT NO. 7
; EXPRESSION ERROR.
;
          LDA     0,SEV97
          STA     0,AGG97
          JSRS    ERE26
AGG97:    .BLK    1
;
; SET AC 0 AND AC 1 TO 0.
;
```

```
E~197:  SUB     0,0
        SUB     1,1
        JMP     EN097

;
; ADDITION SYMBOL.SET ADDRESS.
;
ADD97:  LDA     1,ADD98
        STA     1,ADR97
        JMP     GVL97
;
; SUBTRACTION SYMBOL.SET ADDRESS.
;
SUB97:  LDA     1,SUB98
        STA     1,ADR97
        JMP     GVL97
;
; MULTIPLICATION SYMBOL,SET ADDRESS.
;
MUL97:  LDA     1,MPY72
        STA     1,ADR97
        JMP     GVL97
;
; DIVISION SYMBOL,SET ADDRESS.
;
DIV97:  LDA     1,DIV98
        STA     1,ADR97
;
; CALL ON ROUTINE TO CONVERT NEXT OPERAND.
;
GVL97:  JSRS    CVT96
;
; AC 0 CONTAINS BREAK CHARACTER .
; AC 1 CONTAINS VALUE.
; ACC 0 CONTAINS NULL IF EXP CANNOT BE EVALUATED.
;
        STA     0,BRK97
;
; PERFORM OPERATION.
; ACC 0 CONTTAINS FINAL RRESULT.
; ACC 0 AND ACC 1 ARE INPUTS NEEDED BY ROUTINE.
;
        LDA     0,VAL97
        JSRS    ACR97
        STA     0,VAL97
        JMP     LBL97
;
; CH. IS BLANK,END OF EXPN.
```

```
; BREAK CH. IS IN ACC 0,RETURN.
;
NXT97:  LDA     1,/AL97
ENC97:  LDA     2,AC297
        JMPS    AC397
;
;
;*****************************************************
;
; NAME.                         C/T96.
; FUNCTION.                     EVALUATES THE NEXT
;                               OPERAND IF POSSIBLE.
; INPUT.                        CHPTR POINTING TO
;                               CH. BEFORE SYMBOL/NUMBER.
; OUTPUT.                       CHPTR POINTS TO BREAK.
;                               VALUE IN ACC 1.
;                               BREAK IN ACC 0,NULL
;                               IF ERROR.
; CALLS.                        SYC97,SKIP3,DEC91,
;                               HEX94,OCT92,SYM95.
; CALLED-BY.                    E/L97.
; GLOBAL-VARIABLES-USED.        NONE
; GLOBAL-VARIABLES-CHANGED.     NONE.
; ERROR-BITS-SET.               NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC296:  .BLK    1
AC396:  .BLK    1
;
; CONSTANTS.
;
PER96:          45              ; %
ATT96:          100             ; @
DOL96:          44              ; $
BLK96:          40              ; BLANK
CHZ96:          60              ; CH "0
```

```
CH9696:         71                      ;CH 9
COM696:         54                      ; ,
;
; SAVE ACS CONTENTS,ENTRY POINT
;
CV196:  STA     2,AC296
        STA     3,AC396
;
; GET FIRST CHARACTER OF NUMBER-SYMBOL AND DECIDE
; WHICH ROUTINE TO CALL.
;
        JSR@    GNC96
        LDA     1,PER96
        SUBZ    0,1,SNR
        JMP     BIN96                   ; CH IS %
;
        LDA     1,COL96
        SUBZ    0,1,SNR
        JMP     HEX96                   ; CH IS $
;
        LDA     1,ATT96
        SUBZ    0,1,SNR
        JMP     OCT96                   ; CH IS @
;
; CHECK IF CH IS BLANK OR COMMA
;
        LDA     1,BLK96
        SUBZ    0,1,SNR
        JMP     NXT96                   ; CH IS BLANK
;
        LDA     1,COM96
        SUBZ    0,1,SNR
        JMP     NXT96                   ; CH IS COMMA
;
; CHECK IF DECIMAL OR SYMBOL
;
        LDA     1,CHZ96
        LDA     2,CH196
        ADCZE   2,0,SNC
        ADCZE   0,1,SZC
        JMP     SYM96                   ; IT ISA SYMBOL
;
; CH IS BETWEEN 0-9,TRANSFER TO CONVERSION ROUTINE
; AFTER ADJUSTING CHPTR.
;
        DSZ     CHPTR
        JSR@    DEC91
```

```
          JMP     END96
;
; CH IS BINARY,TRANSFER TO ROUTINE.
;
BIN96:    JSR$    BNY93
          JMP     END96
;
; CH IS HEXADECIMAL.
;
HEX96:    JSR$    HEX94
          JMP     END96
;
; CH IS OCTAL.
;
OCT96:    JSR$    OCT92
          JMP     END96
;
; CH IS SYMBOL.
;
SYM96:    DSZ     CHPTR
          JSR$    SYM95
          JMP     END96
;
; CH IS BLANK OR COMMA,,SET VALUE TO 0
;
NXT96:    SUB     1,1
;
; RESTORE ACC CONTENTS,RETURN
;
END96:    LDA     2,AC296
          JMP$    AC396
;
;*****************************************************
;
; NAME.                         MPY/2.
; FUNCTION.                     MULTIPLY TWO NUMBERS.
;                               ACC 0 = ACC 0 * ACC 1.
; INPUT.                        ACC 0,ACC 1.
; OUTPUT.                       ACC 0.
; CALLS.                        NONE.
; CALLED-BY.
```

```
;
;  GLOBAL-VARIABLES-USED.                    EVL97.
;  GLOBAL-VARIABLES-CHANGED.                 NONE.
;  ERROR-BITS-SET.                           NONE.
;                                            NONE.
;
;  SPACE FOR SAVING ACC CONTENTS.
;
ACC72:    .BLK    1
AC372:    .BLK    1
;
;  CONSTANTS.
;
CNS72:            -20                       ; DEC  -16
;
;  ENTRY POINT,SAVE ACC CONTENTS.
;
MP172:    STA     3,AC372
          STA     2,AC272
;
          SUBC    2,2
          LDA     3,CNS72                   ;16 TIMES THRO
;                                           THE LOOP.
;
;
LBL72:    MOVR    0,0,SNC                   ;CHECK NEXT
;                                           MULTIPLIER BIT.
          MOVR    2,2,SKP                   ;0,SKIP.
          ADDZR   1,2                       ; 1,ADD AND
;                                           SHIFT.
          INC     3,3,SZR                   ; CHECK FOR 16
;                                           TIME THRO LOOP.
          JMP     LBL72
          MOVCR   0,0                       ; YES,SHIFT
;                                           LAST BIT.
;
;  RETURN.
;
          LDA     2,AC272
          JMPS    AC372
;
;  ********************************************* ***-***
;
```

```
;NAME.                          IMM99.
;FUNCTION.                      TAKES CARE OF IMMEDIATE
;                               MODE OF ADDRESSING.
;INPUT.                         CHPTR POINTS TO FIRST CH.
;                               OF EXPRESSION.
;OUTPUT.                        CHPTR UNCHANGED IF NOT IMM.
;                               CHPTR SET TO BREAK IF IMM.
;                               LENTH SET ,WDPCE SET
;                               ACC 0 = 0,IF IMM,
;                                     =1,OTHERWISE.
;CALLS.                         GCH31.
;CALLED-BY.                     LOTS.
;GLOBAL-VARIABLES-USED.         WDPCE.
;GLOBAL-VARIABLES-CHANGED.      WDPC1,LENTH.
;ERROR-BITS-SET.                NONE.

;SPACE FOR SAVING ACCS.

AC199:   .BLK    1
AC299:   .BLK    1
AC399:   .BLK    1

;CONSTANT

CH299:           2
NLF99:           43                     ; CH =
LIT99:           47                     ; CH "
MSK99:           377                    ; MASK

;SAVE ACC CONTENTS.

IM199:   STA     1,AC199
         STA     2,AC299
         STA     3,AC399

;CHECK IF CH IS =
```

```
          JSR<      GCH81
          LDA       1,NL099
          SUBE      0,1,SZR
          JMP       NXT99
;
;  CH IS E.CHECK IF NEXT IS *
;
          JSR<      GNC87
          LDA       1,LIT99
          SUBE      0,1,SZR
          JMP       EXP99
;
;  CH. IS A LITERAL,STORE IN WSPCE AS IT IS.
;
          JSR<      GNC87
          STA<      0,WSPCE
          MOVS      0,1
          JSR<      GNC87
          ADD       0,1
          LDA       3,WSPCE
          STA       1,1,3
          SUB       0,0
          LDA       1,CH239
          STA       1,LENTH
          JMP       END99
;
;   CH IS NOT LITERAL,EVAL EXPRESSION.
;
EXP99:    JSR<      EVL97
          LDA       0,MSK99
          ANDZ      1,0
          STA<      0,WSPCE
          LDA       3,WSPCE
          STA       1,1,3
          SUB       0,0
          LDA       1,CH239
          STA       1,LENTH
          JMP       END99
;
;  SET TO 1 ACC 0
;
NXT99:    SUB       0,0
          INC       0,0
;
;  RESTORE ACC CONTENTS.
;
END99:    LDA       1,AC199
```

```
        LDA     2,AC299
        JMPS    AC399
;
;
;  ****************************************************
;
;  NAME.
;                                  BCC70.
;  FUNCTION.
;                                  SERVICES THE RELATIVE MODE
;                                  OF ADDRESSING,FORMAT 9.
;                                  INSTRUCTIONS SERVICED ARE
;                                  BCC BCS BEQ BGE BGT BHI
;                                  BLE BLS BLT BMI BNE BPL
;                                  BRA BSR BVC BVS.
;  INPUT.
;                                  CHPTR POINTS TO CH. AFTER
;                                  OP-CODE.
;  OUTPUT.
;                                  LENTH,WSPCE SET.
;                                  CHPTR POINTS TO BREAK.
;  CALLS.
;                                  EVL97,ENE26.
;  CALLED-BY.
;                                  SPM25.
;  GLOBAL-VARIABLES-USED.
;                                  WSPCE,LOCTR,BSVAL,ABFLG.
;  GLOBAL-VARIABLES-CHANGED.
;                                  LENTH,WSPC1.
;  ERROR-BITS-SET.
;                                  10(F).
;
;
;  SPACE FOR SAVING ACC CONTENTS.
;
AC070:    .BLK     1
AC170:    .BLK     1
AC270:    .BLK     1
AC370:    .BLK     1
;
;  CONSTANTS.
;
CN270:             2
CN370:             3
CN470:             12
MSK70:             377
;
```

```
;
; SAVE ACC CONTENTS.
;
BC17C:  STA     0,ACC7C
        STA     1,AC17C
        STA     2,AC27C
        STA     3,AC37C
;
; IF ABFLG IS SET,PROCEED
; IF ABFLG = 3 ,PROCEED,OTHERWISE FORMAY ERROR.
;
        JSR$    SCP23
        JMP     END7C
        LDA     0,ABFLG
        LDA     1,CN37C
        SUBZ    0,1,SZR
        JMP     ERR7C
;
; EVALUATE EXP
;
        JSR$    EVL97
;
; SET LENTH
;
        LDA     2,CN27C
        STA     2,LENTH
;
; GET VALUE
;
        LDA     2,LOCTR
        SUB     2,1
        LDA     2,CN27C
        SUB     2,1
        LDA     2,MSK7C
        AND     1,2
        LDA     1,BS/AL
        MOVS    1,1
        ADD     1,2
        STA$    2,WSPCE
        JMP     END7C
ERR7C:  LDA     0,CN47C
        STA     0,AGC7C
        JSR$    ERR26
AGC7C:  .BLK    1
;
; RESTORE ACC,RETURN
;
END7C:  LDA     0,ACC7C
```

```
        LDA     1,AC170
        LDA     2,AC270
        JMPS    AC370
;
;
; ************************************************************
;
; NAME.
;                                EX602.
; FUNCTION.
;                                SERVICES EXTENDED MODE OF
;                                ADDRESSING.
; INPUT.
;                                CHPTR POINTS TO FIRST CH.
;                                OF EXPRESSION.
; OUTPUT.
;                                CHPTR POINTS TO BREAK.
;                                LENTH SET TO 3.
;                                WSPCE BYTES 2 AND 3 SET.
; CALLS.
;                                EVL97.
; CALLED-BY.
;                                LOTS.
; GLOBAL-VARIABLES-USED.
;                                WSPCE.
; GLOBAL-VARIABLES-CHANGED.
;                                WSPC1,LENTH.
; ERROR-BITS-SET.
;                                NONE.
;
;
; SPACE FOR SAVING ACC CONTENTS.
;
A0602:  .BLK    1
A1602:  .BLK    1
A2602:  .BLK    1
A3602:  .BLK    1
;
; CONSTANTS.
;
C3602:          3
MK602:          377
;
; SAVE ACC CONTENTS.
;
E1602:  STA     0,A0602
        STA     1,A1602
```

```
        STA     2,A2800
        STA     3,A3800

;EVALUATE EXP .

        JSRS    EVLS7

;SET LENTH

        LDA     2,C3800
        STA     2,LENTH

;SET MSPCE

        LDA     3,MSPCE
        LDA     2,MK800
        ANDS    1,2
        STA     2,1,3
        LDA     2,MK800
        IOVS    1,1
        ADC     1,2
        STA     2,0,3

;RESTORE ACC CONTENTS.

        LDA     0,A0800
        LDA     1,A1800
        LDA     2,A2800
        JMPS    A3800

;*****************************************************

;NAME.                          DIRCT.

;FUNCTION.                      SERVICES THE DIRECT MODE
                                OF ADDRESSING.

;INPUT.                         CHPTR POINTS TO FIRST CH.
                                OF EXPRESSION.

;OUTPUT.                        CHPTR POINTS TO BREAK.
                                LENTH SET ,MSPCE SET.

;CALLS.                         EVLS7.

;CALLED-BY.
```

```
;
;  GLOBAL-VARIABLES-USED.                    LOTS.
;                                            NSPCE,DIRAD,PSFLG,DEPT,
;                                            LNCTR.
;  GLOBAL-VARIABLES-CHANGED.                 DIRA1,NSPC1,LENTH.
;  ERROR-BITS-SET.                           NONE.
;
;  SPACE FOR SAVING ACC CONTENTS.
;
AC0600:    .BLK    1
A16600:    .BLK    1
A26600:    .BLK    1
A36600:    .BLK    1
;
;  CONSTANTS.
;
C17700:            377
C26600:            2
C36600:            3
C46600:            62                        ; DEC 50
C96600:            177400
VL8600:    .BLK    1
;
;  SAVE ACC CONTENTS.
;
D16600:    STA     0,AC0600
           STA     1,A16600
           STA     2,A26600
           STA     3,A36600
;
;  EVALUATE  EXPRESSION.
;
           JSR@    EVL97
           STA     1,VL8600
;
;  AC 1 CONTAINS VALUE,AC 0 CONTAINS BREAK.
;  AC 0 = 0 IF ERROR IN EVALUATION.
;
           MOV     0,0,SNR
           JMP     EX8600
;
           LDA     0,PSFLG
           LDA     2,C26600
           SUBE    0,2,SNR
```

```
          JMP     P2000
;
; PASS = 1,CHECK IF DIRAD IS FULL,AC 1 CONTAINS VALUE
;
          LDA     0,DIRAD
          LDA     2,C4000
          ADD     0,2
          LDA     0,DREPT
          SUBE    2,0,SNR
          JMP     EX000
;
; TABLE NOT FULL,CHECK IF ACC IS BETWEEN    00 AND  FF.
;
          LDA     0,C0000
          ANDE    0,1,SZR
          JMP     EX000
;
; PASS = 1,DIRAD IS NOT FULL ,VALUE LESS THAN FF.
;
          LDA     0,LNGTH
          STA     0,DREPT
          ISZ     DREPT
          JMP     DR000
;
; PASS TO 2
; CHECK IF LINE PRESENT IN DIRAD,IF SO DIRECT ELSE EXTENDED.
;
P2000:    LDA     2,DIRAD
          LDA     3,DREPT
          LDA     0,LNGTH
;
LB000:    SUBE    2,3,SNR
          JMP     EX000                   ; NOT FOUND
          LDA     1,0,2
          SUBE    0,1,SNR
          JMP     DR000                   ; FOUND
          INC     2,2
          JMP     LB000
;
; ENTER WORD,SETLENTH,STORE VALUE.
;
EX000:    LDA     0,C3000
          STA     0,LENTH
          LDA     2,WSPCE
          LDA     1,VL000
          LDA     0,C1800
```

```
        ANDS    1,0
        STA     0,1,2
        LDA     0,C1800
        MOVS    1,1
        AND     1,0
        STA     0,0,2
        JMP     EN800
;
; DIRECT MODE,SET LENTH ,STORE VALUE
;
DR800:  LDA     1,VL800
        LDA     0,C1800
        AND     1,0
        STA$    0,WSPCE
        LDA     0,C2800
        STA     0,LENTH
        ISZ     A3800
;
; RESTORE ACC AND RETIRN.
;
EN800:  LDA     0,A0800
        LDA     1,A1800
        LDA     2,A2800
        JMP$    A3800
;
;
;  *********************************************************
;
; NAME.
;                         IN801.
; FUNCTION.
;                         SERVICES THE INDEXED
;                         MODE OF ADDRESSING.
; INPUT.
;                         CHPTR POINTS TO FIRST
;                         CHARACTER OF EXPN.
; OUTPUT.
;                         CHPTR POINTS TO BREAK.
;                         IF INDEXED,CHPTR POINTS TO
;                         BREAK,ACC 0 = 0,LENTH SET
;                         TO 2,WSPCE SET.
;                         IF NOT INDEXED ACC 0 = 1,
;                         CHPTR UNCHANGED.
; CALLS.
;                         GCH01,GNC87,EVL87,GL341.
; CALLED-BY.
;                         LOTS.
;
```

```
; GLOBAL-VARIABLES-USED.                    NSPCE.
; GLOBAL-VARIABLES-CHANGED.                 LENTH,NSPC1,CHPTR.
; ERROR-BITS-SET.                           NONE.


; SPACE FOR SAVING ACC.

A1SC1:    .BLK    1
A2SC1:    .BLK    1
A3SC1:    .BLK    1
TMSC1:    .BLK    1

; CONSTANTS.

C1SC1:            121
C2SC1:            2
C3SC1:            26133                   ; CMS.    ≠,X≠
C4SC1:            54340                   ;  CMS   ≠X ≠
MKSC1:            377

; SAVE ACC CONTENTS,ENTRY

I1SC1:    STA     1,A1SC1
          STA     2,A2SC1
          STA     3,A3SC1

; CHECK IF ERROR

          LDA     0,CHPTR
          LDA     1,C1SC1
          SUBE    0,1,SNR
          JMP     NISC1

; CHECK IF CH IS X

          JSR@      GCH+1
          MOVZ      0,2
          JSR@    GNC87
          ADC     0,2
          LDA     1,C4SC1
          SUB     2,1,SNR
          JMP     LESC1

;
```

```
;
; GET NEXT NON-BLANK CHARACTER.
;
          DSZ       CHPTR
          LDA       0,CHPTR
          STA       0,TM801
          JSR@      GLB41
          DSZ       CHPTR
          JSR@      GCH81
          MOV       0,1
          DSZ       CHPTR
          JSR@      GCH81
          MOVS      0,0
          ADC       0,1
;
; CHECK IF #,XA.
;
          LDA       0,C3801
          SUBE      0,1,SZR
          JMP       NI801
;
;
; INDEXED.
; EVALUATE EXP AND SET LENGTH,AC 0,WSPCE
;
IDEC1:    LDA       0,TM801
          STA       0,CHPTR
          JSR@      EVL97
LBEC1:    LDA       2,C2801
          STA       2,LENTH
          LDA       2,MK801
          AND       1,2
          STA@      2,WSPCE
          SUB       0,0
          JMP       EN801
;
; NOT INDIRECT,SET AC 0
;
NIEC1:    SUB       0,0
          INC       0,0
          LDA       1,TM801
          STA       1,CHPTR
;
; RESTORE ACC AND RETURN.
;
ENEC1:    LDA       1,A1801
          LDA       2,A2801
          JMP@      A3801
```

```
;*****************************************************
;
;NAME.                          SAB98.
;FUNCTION.                      SETS ABFLG IF IT IS NOT
;                               ALREADY SET.
;                               IE. IF ABFLG = 3,THEN
;                               SET IT TO EITHER 2 OR 1.
;INPUT.                         CHPTR POINTS TO CH. AFTER
;                               OP-CODE.
;OUTPUT.                        ABFLG SET TO 2 OR 3,
;                               CHPTR POINTS TO FIRST CH.
;                               OF EXPRESSION.
;                               RETURNS TO ACC 3 IF ERROR.
;                               RETURNS TO ACC 3 + 1
;                               OTHERWISE.
;CALLS.                         GLB41,SCP83.
;CALLED-BY.                     LOTS.
;GLOBAL-VARIABLES-USED.         NONE.
;GLOBAL-VARIABLES-CHANGED.      ABFLG.
;ERROR-BITS-SET.                1,(F).
;
;SPACE FOR SAVING ACC CONTENTS.
;
AC098:    .BLK    1
AC198:    .BLK    1
AC298:    .BLK    1
AC398:    .BLK    1
;
;CONSTANTS.
;
CN098:            12                    ;DEC 10
CN298:            2
CN198:            1
CN598:            121                   ;DEC 81
CNA98:            40440                 ;ASCII
```

```
CNB98:              41040                  ;#B #
;
; ENTRYY POINT,SAVE ACC CONTENTS.
;
SA198:    STA       0,AC098
          STA       1,AC198
          STA       2,AC298
          STA       3,AC398
;
; SEE IF ABFLG IS 2 OR 1
;
          LDA       0,ABFLG
          LDA       1,CN298
          SUBZ      0,1,SNR
          JMP       SET98
;
          LDA       1,CN198
          SUBZ      0,1,SNR
          JMP       SET98
;
; ABFLG NOT SETT.SET IT. CALL SCP23 TO SET CHPTR TO
; NEXT NON BLANK CHARACTER.
;
          JSRS      SCP23
          JMP       END98                  ; ERROR.
;
;
          JSRS      GCH61
          MOVS      0,1
          JSRS      GNC67
          ADD       1,0
          LDA       1,CNA98
          SUBZ      0,1,SNR
          JMP       OKA98
;
;
          LDA       1,CNB98
          SUBZ      0,1,SNR
          JMP       OKB98
;
; ERROR(FORMAT),BIT NO. 10 SET.
;
          LDA       0,CN998
          STA       0,AG098
          JSRS      ENE26
AG098:    .BLK      1
          JMP       END98                  ;ERROR RETURN
```

```
;
;  CH IS #A #
;
OKA98:    LDA     1,CH198
          STA     1,ABFLG
          ISZ     AC398
          JSR<    SCP23
          JMP     END98
          JMP     END98
;
;  CH IS #B #
;
OKB98:    LDA     1,CH298
          STA     1,ABFLG
          ISZ     AC398
          JSR<    SCP23
          JMP     END98
          JMP     END98
;
;  SET CHPTR
;
SET98:    JSR<    GLB41
          JSR<    SCP23
          JMP     END98
          ISZ     AC398
;
;  RESTORE ACC AND RETURN.
;
END98:    LDA     0,ACC98
          LDA     1,AC198
          LDA     2,AC298
          JMP<    AC398
;
;  ********************************************************
;
;  NAME.                          ACC98.
;  FUNCTION.                      ACC 0 = ACC 0 + ACC 1.
;  INPUT.                         ACC 0,ACC 1.
;  OUTPUT.                        ACC 0.
;  CALLS.                         NONE.
;  CALLED-BY.
```

```
                                EVL97.
GLOBAL-VARIABLES-USED.          NONE.
GLOBAL-VARIABLES-CHANGED.       NONE.
ERROR-BITS-SET.                 NONE.

AC188:  ADC     1,1
        JMP     0,3

*****************************************************

NAME.                           SUB89.
FUNCTION.                       ACC 0 = ACC 0-ACC 1.
INPUT.                          ACC 0,ACC 1.
OUTPUT.                         ACC 0.
CALLS.                          NONE.
CALLED-BY.                      EVL97.
GLOBAL-VARIABLES-USED.          NONE.
GLOBAL-VARIABLES-CHANGED.       NONE.
ERROR-BITS-SET.                 NONE.

SU189:  SUB     1,1
        JMP     0,3

*****************************************************

NAME.                           DIV90.
FUNCTION.                       ACC 0 = ACC 0 / ACC 1.
INPUT.                          ACC 0,ACC 1.
```

```
; OUTPUT.                          ACC 0.
; CALLS.                           ENE26.
; CALLED-BY.                       EVL97.
; GLOBAL-VARIABLES-USED.           NONE.
; GLOBAL-VARIABLES-CHANGED.        NONE.
; ERROR-BITS-SET.                  3 (V).
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC290:   .BLK    1
AC390:   .BLK    1
;
; CONSTANTS
;
CNS90:           -20                        ; DEC -16
CNS90:           10                         ;DEC 8
;
; ENTRY POINT,SAVE ACC
;
DI190:   STA     2,AC290
         STA     3,AC390
;
; CHECK ACC 1 IF IT IS 0
;
         MOV     1,1,SNR
         JMP     ERR90                      ; OVERFLOW ERROR
;
; ACC 1 IS NON ZERO,CLEAR ACC 2,SET UP COUNTER FOR 16.
;
         SUB     2,2
         LDA     3,CNS90
         MOVZL   0,0                        ;SHIFT LOW
                                            DIVIDEND.
;
;
;
LBL90:   MOVL    2,2                        ; SHIFT HIGH
                                            DIVIDEND.
         SUBE    1,2,SZC                    ; DOES DIVIDEND
                                             GO IN.
;
```

```
        SUB     1,2                     ;YES
        MOVL    0,0
        INC     3,3,SZR
        JMP     LBL90
;
; DONE,RETURN
;
        JMP     END90
;
; SET OVERFLOW ERROR,SET AC 0.
;
ERR90:  LDA     0,ON390
        STA     0,AG190
        JSR$    ERE25
AG190:  .BLK    1
;
        SUB     0,0
;
; RESTORE ACC CONTENTTS.
;
END90:  LDA     2,AC290
        JMP$    AC390
;
;
; **********************************************************
;
; NAME.                         ADC63.
;
; FUNCTION.                     SERVICED THE INSTNS.
;                               WITH FORMAT NO. 1.
;                               THE INSTNS. SERVICED ARE
;                                ADC ADD AND BIT CMP
;                                EOR LDA ORA SBC SUB.
; INPUT.
;                               CHPTR POINTS TO CH. AFTER
;                               OP-CODE.
; OUTPUT.
;                               WSPCE,LENTH SET.
;                               CHPTR POINTS TO END OF EXPN
; CALLS.
;                               SA866,INM99,DI600,IN301.
; CALLED-BY.
;                               SPM25.
; GLOBAL-VARIABLES-USED.
;                               WSPCE,ABFLG,BSVAL.
; GLOBAL-VARIABLES-CHANGED.
```

```
;
;  ERROR-BITS-SET.                    MSPC1,LENGTH.
;                                     NONE.
;
;
;  SPACE FOR SAVING ACC CONTENTS.
;
AC063:    .BLK     1
AC163:    .BLK     1
AC263:    .BLK     1
AC363:    .BLK     1
;
;   CONSTANTS.
;
FOR63:             4
TWO63:             2
;
;  SAVE ACC,ENTRY
;
AD163:    STA      0,AC063
          STA      1,AC163
          STA      2,AC263
          STA      3,AC363
;
;  CALL ON ROUTINE TO SET CHPTR,ABFLG.
;
          JSRS     SA399
          JMP      END63                    ; ERROR.
;
;  CHECK WHICH OF THE FOUR ADDRESSING TYPES .
;
          SUB      1,1
          JSRS     IM199
          MOV      0,0,SNR
          JMP      IM163
;
;
          JSRS     IN001
          MOV      0,0,SNR
          JMP      IN063
;
;
          JSRS     DIR00
          JMP      EXT63                    ; EXTENDED MODE
          JMP      DIR63
EXT63:    INC      1,1
IN063:    INC      1,1
```

```
DIF63:   INC     1,1
;
; CHECK WHETHR A OR B.
;
INR63:   LDA     0,ABFLG
         LDA     2,FOR63
         LDA     3,TWO63
         SUBE    0,3,SNR
         ADC     2,1
;
; FORM FIRST BYTE.
;
         MOVZL   1,1
         MOVZL   1,1
         MOVZL   1,1
         MOVZL   1,1
         LDA     0,BSVAL
         ANDCS   0,1
         LDAS    0,WSPCE
         ADD     1,0
         STAS    0,WSPCE
;
; RETURN.
;
END63:   LDA     0,AC063
         LDA     1,AC163
         LDA     2,AC263
         JMPS    AC363
;
;*****************************************************
;
; NAME.                          RMB61.
; FUNCTION.                      SERVICES THE PSEUDO-OP RMB.
;                                ALLOCATES STORAGE SPACE.
; INPUT.                         NONE.
; OUTPUT.                        NONE.
; CALLS.                         STR64,EVL97,ERR26.
; CALLED-BY.                     SP#25.
; GLOBAL-VARIABLES-USED.         NONE.
```

```
;
; GLOBAL-VARIABLES-CHANGED.                VALUE,VLFLG,PNFLG.
; ERROR-BITS-SET.                          11(F).
;
; SPACE FOR SAVING ACS CONTENTS
; ON ENTRY CHPTR IS SET TO CH AFTER OPCODE
;
AC061:    .BLK    1
AC161:    .BLK    1
AC261:    .BLK    1
AC361:    .BLK    1
;
; CONSTANTS
;
EVN61:            13
;
; SAVE ACS ,ENTRY
;
FN161:    STA     0,AC061
          STA     1,AC161
          STA     2,AC261
          STA     3,AC361
          ISZ     VLFLG
;
; SET CHPTR AND EVALUATE
;
          JSRS    ST304
          JMP     END61
          JSRS    EVL97
;
; CHECK IF IT CAN BE EVALUATED
;
          MOVS    0,0,SNR
          JMP     ERR61
;
; SET LOCTR AND PNFLG
;
          STA     1,VALUE
          LDA     0,LOCTR
          ADD     1,0
          STA     0,LOCTR
;
          ISZ     PNFLG
          JMP     END61
```

```
;SET BIT 11
;
EVA61:  LDA     0,EV061
        STA     0,ARG61
        JSRS    ENE26
ARG61:  .BLK    1
;
;  RESTORE ACC AND RETURN
;
EVC61:  LDA     0,AC061
        LDA     1,AC161
        LDA     2,AC261
        JMPS    AC361
;
;*****************************************************
;
;NAME.                          OPG59.
;FUNCTION.                      THIS ROUTINE SERVICES THE
;                               PSEUDO-OP ORG.SETS THE
;                               PROGRAM COUNTER.
;INPUT.                         NONE.
;OUTPUT.                        NONE.
;CALLS.                         STO04,EVL07,ENE26.
;CALLED-BY.                     SPM25.
;GLOBAL-VARIABLES-USED.         LBFLG.
;GLOBAL-VARIABLES-CHANGED.      LOCTR,LOCTN,PNFLG.
;ERROR-BITS-SET.                12(P),11(R).
;
;SPACE FOR SAVING ACC CONTENTS.
;
AC359:  .BLK    1
AC159:  .BLK    1
AC259:  .BLK    1
AC059:  .BLK    1
;
```

```
; CONSTANTS.
;
TWL59:          14
ONE59:          1
EVN59:          13
;
; SAVE ACCS. ENTRY
;
OPI59:  STA     0,AC059
        STA     1,AC159
        STA     2,AC259
        STA     3,AC359
;
; CHECK IF LABEL IS PRESENT
;
        LDA     0,LBFLG
        LDA     1,ONE59
        SUBE    0,1,SNR
        JMP     ER259
;
; SET CHPTR AND EVALUATE EXP.
;
        JSRS    STS64
        JMP     END59
        JSRS    EVL97
        MOV     0,0,SNP
        JMP     ER559
;
        STA     1,LOCTR
        STA     1,LOCTM
        ISZ     PNFLG
        JMP     END59
;
; ERROR,SET BIT NO. 12
;
ER259:  LDA     0,TWL59
        STA     0,AG259
        JSRS    ERE26
AG259:  .BLK    1
        JMP     END59
;
; SET ERROR BIT NO. 11(R).
;
ER559:  LDA     0,EVN59
        STA     0,ARG59
        JSRS    ERE26
ARG59:  .BLK    1
```

```
END59:  LDA     0,AC059
        LDA     1,AC159
        LDA     2,AC259
        JMPS    AC359
;
; ************************************************************
;
; NAME.                                 EQU53.
; FUNCTION.                             SERVICES THE PSEUDO-OP EQU.
;                                       SETS SYMBOL VALUE.
; INPUT.                                NONE.
; OUTPUT.                               NONE.
; CALLS.                                ST034,EVL07.
; CALLED-BY.                            SAR25.
; GLOBAL-VARIABLES-USED.                LDFLG,ABFLG.
; GLOBAL-VARIABLES-CHANGED.             VLFLG,LDFLG,VALUE.
; ERROR-BITS-SET.                       B(N).
;
;
;  SPACE FOR AVING ACS CONTENTS.
;
AC053:  .BLK    1
AC153:  .BLK    1
AC253:  .BLK    1
AC353:  .BLK    1
;
; CONSTANTS
;
MIN53:          11                      ; DEC 9.
ONE53:          1
;
; SAVE ACSS ,ENTRY
;
EQ153:  STA     0,AC053
        STA     1,AC153
        STA     2,AC253
        STA     3,AC353
```

```
          ISZ       VLFLG
          SUB       1,1
          STA       1,LCFLG
;
; CHECK IS THERE IS LABEL.IF SO OKAY.
;
          LDA       3,LBFLG
          MOV       3,0,SNR
          JMP       ERR53
;
; SET CHPTR AND EVALUATE
;
          JSRS      STO04
          JMP       END53
          JSRS      EVL97
;
; CHECK IF EVALUATED PROPERLY
;
          MOV       0,3,SNR
          JMP       IMP53
;
;  EVALUATION IS OKAY,SET VALUE
;
          LDA       3,ADLBL
          STA       1,3,3
          STA       1,VALUE
          JMP       END53
;
; SET BIT NO. 9.
;
ERR53:    LDA       3,MIN53
          STA       3,ARG53
          JSRS      ERE26
ARG53:    .BLK      1
          JMP       END53
;
; EXPR. CANNOT BE EVALUATED,SET FIRST NO TO ZERO.
;
IMP53:    LDA       3,ADLBL
          SUB       0,0
          STA       0,0,3
;
; RESTORE ACC AND RETURN
;
END53:    LDA       0,AC053
          LDA       1,AC153
          LDA       2,AC253
```

```
        JMPS    AC353
;
; *************************************************************
;
; NAME.                             OER14.
; FUNCTION.                         OR THE ERROR WORDS
;                                   P1ERR,PRERR.
; INPUT.                            NONE.
; OUTPUT.                           NONE.
; CALLS.                            ORR77.
; CALLED-BY.                        PASS2.
; GLOBAL-VARIABLES-USED.            P1ERR.
; GLOBAL-VARIABLES-CHANGED.         PRERR.
; ERROR-BITS-SET.                   NONE.
;
; SPACE FOR ACC CONTENTS
AC314:  .BLK    1
; ENTRY POINT
OE114:  STA     3,AC314
;
        LDA     3,P1ERR
        STA     3,AG114
        LDA     3,PRERR
        STA     3,AG214
        JSRS    ORR77
AG114:  .BLK    1
AG214:  .BLK    1
AG314:  .BLK    1
        LDA     3,AG314
        STA     3,PRERR
;
; RETURN
;
```

```
        JMPS    AC314
;
;*****************************************************************
; NAME.                          STA64.
; FUNCTION.                      SERVICES THE INSTNS.
;                                WITH FORMAT NO. 1.
;                                IE. STA.
; INPUT.
; OUTPUT.                        CHPTR POINTS TO BREAK.
;                                WSPCE,LENTH SET.
; CALLS.                         SAB98,DIB00,INC01.
; CALLED-BY.                     SPM25.
; GLOBAL-VARIABLES-USED.         BSVAL,ABFLG,WSPCE.
; GLOBAL-VARIABLES-CHANGED.      WSPC1.
; ERROR-BITS-SET.                NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
AC064:  .BLK    1
AC164:  .BLK    1
AC264:  .BLK    1
AC364:  .BLK    1
;
; CONSTANTS.
;
TWO64:          2
FOR64:          4
;
; SAVE ACC CONTENTS,ENTRY.
;
ST164:  STA     0,AC064
        STA     1,AC164
        STA     2,AC264
        STA     3,AC364
;
; CALL ON ROUTINE TO SET CHPTR AND ABFLG.
;
```

```
          JSR@    SABY6
          JMP     END64
;
; CHECK WHICH MODE AND SET ACC 1 ACCORDINGLY.
;
          SUB     1,1
          JSR@    INB01
          MOV     0,J,SNR
          JMP     IN064
          JSR@    DIB00
          JMP     EXT64
          JMP     DIR64
;
EXT64:    INC     1,1
IN064:    INC     1,1
;
; CHECK WHETHER ACC A OR ACC B.
;
DIR64:    LDA     0,ABFLG
          LDA     2,FOR64
          LDA     3,TWO64
          SUBE    0,3,SNR
          ADD     2,1
;
; FORM FIRST BYTE.
;
          MOVLZ   1,1
          MOVLZ   1,1
          MOVLZ   1,1
          MOVLZ   1,1
          LDA     0,BSVAL
          ADDCS   0,1
          LDA@    0,WSPCE
          ADD     1,J
          STA@    0,WSPCE
;
; RESTORE ACC AND RETURN.
;
END64:    LDA     0,AC064
          LDA     1,AC164
          LDA     2,AC264
          JMP@    AC364
;
;*****************************************************************
;
; NAME.
```

```
;  FUNCTION.                    ASL65.
;
;                               SERVICES ONE INSTRS. WITH
;                               FORMAT NO. 2.
;                               ASL ASR CLR COM DEC
;                               INC LSR NEG ROL ROR TST.
;  INPUT.
;                               CHPTR POINTS TO CH. AFTER
;                               OP-CODE.
;  OUTPUT.
;                               CHPTR POINTS TO BREAK,
;                               WSPCE,LENTH SET.
;  CALLS.
;                               SCR23,GCHR1,GNC07,
;                               INC01,ERR02.
;  CALLED-BY.
;                               SR025.
;  GLOBAL-VARIABLES-USED.
;                               ADFLG,DSVAL,WSPCE.
;  GLOBAL-VARIABLES-CHANGED.
;                               LENTH,WSPC1.
;  ERROR-BITS-SET.
;                               NONE.

;  SPACE FOR SAVING ACC CONTENTS.

AC065:    .BLK    1
AC165:    .BLK    1
AC265:    .BLK    1
AC365:    .BLK    1

;  CONSTANTS.

CHA65:            40440                 ; #A #
CHB65:            41040                 ;  #B #
ONE65:            1
TWO65:            2

;  SAVE ACC CONTENTS,ENTRY.

AS165:    STA     0,AC065
          STA     1,AC165
          STA     2,AC265
          STA     3,AC365
          LDA     0,ONE65
          STA     0,LENTH
```

```
;
;   CHECK IF ABFLG IS SET
;
        SUB     1,1
;
        LDA     0,ABFLG
        LDA     2,ONE65
        SUBE    0,2,SNR
        JMP     AST65
        LDA     2,TWO65
        SUBE    0,2,SNR
        JMP     BST65
;
; ABFLG IS 3,SEE IF A OR B IS PRESENT.
;
        JSR@    SCP23
        JMP     END65
        JSR@    GCH81
        MOVS    0,2
        JSR@    GNC87
        ADD     2,0
        LDA     2,CHA65
        SUBE    0,2,SNR
        JMP     AST65
        LDA     2,CHB65
        SUBE    0,2,SNR
        JMP     BST65
        DSZ     CHPTR
;
;   IT IS NOT A,B. SEE WHAT TYPE OF ADDRESSING.
;
        JSR@    IN301
        MOV     0,0,SNR
        JMP     IN365
        JSR@    EX302
        INC     1,1
IN365:  INC     1,1
BST65:  INC     1,1
;
; FORM FIRST BYTE
;
AST65:  MOVLZ   1,1
        MOVLZ   1,1
        MOVLZ   1,1
        MOVLZ   1,1
        LDA     0,BSAL
```

```
        ADCS    0,1
        LDAS    0,WSPCE
        ADC     1,0
        STAS    0,WSPCE
;
; RETURN AFTER SAVING ACC CONTENTS.
;
END65:  LDA     0,AC065
        LDA     1,AC165
        LDA     2,AC265
        JMPS    AC365
;
;
; ************************************************************
;
; NAME.
;                                   PSH66.
; FUNCTION.
;                                   SERVICES THE INSTNS.
;                                   PSH,PUL HAVING FORMAT NO.3.
; INPUT.
;                                   CHPTR POINTS TO CH. AFTER
;                                   OP-CODE.
; OUTPUT.
;                                   CHPTR POINTS TO BREAK.
;                                   LENTH,WSPCE SET.
; CALLS.
;                                   SCP23,GCH81,GNC67,END26.
; CALLED-BY.
;                                   SPH25.
; GLOBAL-VARIABLES-USED.
;                                   BSVAL,ASFLG,WSPCE.
; GLOBAL-VARIABLES-CHANGED.
;                                   LENTH,WSPC1.
; ERROR-BITS-SET.
;                                   10(F).
;
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC066:  .BLK    1
AC166:  .BLK    1
AC266:  .BLK    1
AC366:  .BLK    1
;
; CONSTANTS.
;
```

```
CHA66:          40440
CHB66:          41340
ONE66:          1
TWO66:          2
TEN66:          12
;
;  ENTRY,SAVE ACC CONTENTS.
;
PS166:  STA     0,AC066
        STA     1,AC166
        STA     2,AC266
        STA     3,AC366
;
;  SET  LENTH
;
        LDA     0,ONE66
        STA     0,LENTH
        LDA     1,BSVAL
;
;  CHECK WHETHER ABFLG IS SET.
;
        LDA     0,ABFLG
        LDA     2,ONE66
        SUBE    0,2,SNR
        JMP     AST66
        LDA     2,TWO66
        SUBE    0,2,SNR
        JMP     BST66
;
;  ABFLG NOT SET,SEE WHETHER AOR B IS PRESENT.
;
        JSR@    SCP23
        JMP     END66
        JSR@    GCH51
        MOVS    0,2
        JSR@    GNC67
        ADD     2,0
        LDA     2,CHA66
        SUBE    0,2,SNR
        JMP     AST66
        LDA     2,CHB66
        SUBE    0,2,SNR
        JMP     BST66
;
;  ERROR,SET ERROR BIT NO 13
;
        LDA     0,TEN66
```

```
            STA     0,ARG66
            JSRS    ENE26
ARG66:      .BLK    1
            JMP     END66
BST66:      INC     1,1
AST66:      MOVS    1,1
            LDAS    0,WSPCE
            ADD     0,1
            STAS    1,WSPCE
;
; RETURN AFTER RESTORING ACCS.
;
END66:      LDA     0,AC066
            LDA     1,AC166
            LDA     2,AC266
            JMPS    AC365
;
;*****************************************************
;
; NAME.                         CPX67.
; FUNCTION.                     SERVICES THE INSTNS. CPX,
;                               LDS,LDX,HAVING FORMAT NO.4.
; INPUT.                        CHPTR POINTS TO CH. AFTER
;                               OP-CODE.
; OUTPUT.                       CHPTR POINTS TO BREAK.
;                               LENTH,WSPCE SET.
; CALLS.                        ST064,IMM39,IN801,CI300.
; CALLED-BY.                    SPM25.
; GLOBAL-VARIABLES-USED.        WSPCE,BSVAL.
; GLOBAL-VARIABLES-CHANGED.     LENTH,WSPC1.
; ERROR-BITS-SET.               NONE.
;
;    SPACE FOR SAVING ACC CONTENTS.
;
AC067:      .BLK    1
AC167:      .BLK    1
```

```
AC267:    .BLK    1
AC367:    .BLK    1
MSK67:              377
;
;  SAVE ACC CONTENTS,ENTRY
;
CP167:    STA       0,AC067
          STA       1,AC167
          STA       2,AC267
          STA       3,AC367
;
;   CHECK IF ABFLG =3
;
          JSRS      ST804
          JMP       END67
          SUB       1,1
;
;  CHECK WHICH MODE OF ADDRESSING,
;
          JSRS      IMM68
          MOV       0,0,SNR
          JMP       IMM67
          JSRS      IN801
          MOV       0,0,SNR
          JMP       IND67
          JSRS      DI800
          JMP       EXT67
          JMP       DIR67
IMM67:    LDA       3,WSPCE
          LDA       0,1,3
          LDA       2,MSK67
          ANDCS     0,2
          STA       2,1,3
          LDA       2,MSK67
          MOVS      0,0
          AND       0,2
          STAS      2,WSPCE
          ISZ       LENTH
          JMP       NXT67
EXT67:    INC       1,1
IND67:    INC       1,1
DIR67:    INC       1,1
;
;  FORM FIRST BYTE
;
NXT67:    MOVLZ     1,1
          MOVLZ     1,1
```

```
        MOVLZ   1,1
        MOVLZ   1,1
        LDA     0,BSVAL
        ADCS    0,1
        LDAS    0,WSPCE
        ADD     1,0
        STAS    0,WSPCE
;
; RETURN,AFTER RESTORING ACCS.
;
EN067:  LDA     0,AC067
        LDA     1,AC167
        LDA     2,AC267
        JMPS    AC367
;
;
; ****************************************************
;
; NAME.
;                               ST068.
; FUNCTION.
;                               SERVICES THE INSTNS. STS
;                               STX HAVING FORMAT NO.5.
; INPUT.
;                               CHPTR POINTS TO CH. AFTER
;                               OP-CODE.
; OUTPUT.
;                               CHPTR POINTS TO BREAK.
;                               LENTH,WSPCE SET.
; CALLS.
;                               ST604,IN601,SI600.
; CALLED-BY.
;                               SPH25.
; GLOBAL-VARIABLES-USED.
;                               BSVAL,WSPCE.
; GLOBAL-VARIABLES-CHANGED.
;                               WSPC1.
; ERROR-BITS-SET.
;                               NONE.
;
;
; SPACE FOR SAVCNF ACC CONTENTS,ENTRY
;
AC068:  .BLK    1
AC168:  .BLK    1
AC268:  .BLK    1
AC368:  .BLK    1
```

```
;
; SAVE ACC CONTENTS,ENTRY.
;
ST16B:  STA     0,AC.6B
        STA     1,AC16B
        STA     2,AC26B
        STA     3,AC36B
;
; SEE WHETHER AEFLG = 3
;
        JSR$    ST8B4
        JMP     END6B
;
;
        SUB     1,1
;
; CHECK WHICH MODE OF ADDRESSING.
;
        JSR$    IN3B1
        IOV     0,0,SNR
        JMP     IND6B
        JSR$    DI8B0
        JMP     EXT6B
        JMP     DIR6B
;
;
EXT6B:  INC     1,1
IND6B:  INC     1,1
;
; FORM FIRST BYTE
;
DIR6B:  MOVLZ   1,1
        MOVLZ   1,1
        IOVLZ   1,1
        MOVLZ   1,1
;
        LDA     0,BOVAL
        ADCS    0,1
        LDA$    0,WSPCE
        ADC     1,0
        STA$    0,WSPCE
;
;
;   RETURN AFTER RESTORING ACCS.
;
END6B:  LDA     0,AC06B
```

```
        LDA     1,AC169
        LDA     2,AC269
        JMPS    AC368

;*****************************************************

; NAME.                         JMP69.
; FUNCTION.                     SERVICES THE INSTRS. JMP
;                               JSR HAVING FORMAT NO.6.
; INPUT.                        CNPTR POINTS TO CH. AFTER
;                               OP-CODE.
; OUTPUT.                       CNPTR POINTS TO BREAK.
;                               LENTH,WSPCE SET.
; CALLS.                        ST8J4,INCJ1,EXCJ2.
; CALLED-BY.                    SPM25.
; GLOBAL-VARIABLES-USED.        BSVAL,WSPCE.
; GLOBAL-VARIABLES-CHANGED.     WSPC1.
; ERROR-BITS-SET.               NONE.

; SPACE FOR SAVING ACC CONTENTS.

AC069:  .BLK    1
AC169:  .BLK    1
AC269:  .BLK    1
AC369:  .BLK    1

; ENTRY,SAVE ACC CONTENTS.

JM169:  STA     0,AC069
        STA     1,AC169
        STA     2,AC269
        STA     3,AC369

; CHECK ABFLG AND SET CHPTR
```

```
          JSR@    ST304
          JMP     END69
;
; CHECK WHICH MODE
;
          SUB     1,1
          JSR@    IN501
          MOV     0,0,SNR
          JMP     IN069
          JSR@    EX802
          INC     1,1
;
; FORM FIRST BYTE
;
IN069:    MOVLZ   1,1
          MOVLZ   1,1
          MOVLZ   1,1
          MOVLZ   1,1
;
          LDA     0,BSVAL
          ADDS    0,1
          LDA@    0,WSPCE
          ADD     1,0
          STA@    0,WSPCE
END69:    LDA     0,AC069
          LDA     1,AC169
          LDA     2,AC269
          JMP@    AC369
;
;*****************************************************+
;
; NAME.                         ST304.
; FUNCTION.                     CHECKS WHETHER ABFLG = 3.
; INPUT.                        CHPTR POINTS TO CH. AFTER
;                               OP-CODE.
; OUTPUT.                       CHPTR POINTS TO BEG.
;                               OF EXPRESSION.
;                               ACO 3 RETURN IF ERROR.
;                               AC 3 + 1 RETURN OTHERWISE.
; CALLS.
; CALLED-BY.                    SUP23,ENE25.
```

```
;                                       LOTS.
; GLOBAL-/ARIABLES-USED.                ABFLG.
; GLOBAL-/ARIABLES-CHANGED.             ONE.
; ERROR-BITS-SET.                       1)(F).

; SPACE FOR SAVING ACC CONTENTS.

A0804:  .BLK    1
A3804:  .BLK    1

; CONSTANTS.

THR04:          3
TNR04:          12

; SAVE ACC,ENTRY

S1804:  STA     0,A0804
        STA     3,A3804

; CHECK ABFLG=3

        LDA     0,ABFLG
        LDA     3,THR04
        SUBE    0,3,SZR
        JMP     ER804

; SET CMPTR AND RETURN

        JSRS    SCP23
        JMP     ENR04
        ISZ     A3804
        JMP     ENR04

; SET BIT 10

ER804:  LDA     0,THR04
        STA     0,AGR04
        JSRS    ENE26
AGR04:  .BLK    1

; RETURN AFTER RESTORING ACCS.
```

```
;
ENFL4:   LDA     0,A0804
         JMPS    A3304
;
;
;  ************************************************************
;
;  NAME.                         FCC55.
;
;  FUNCTION.                     SERVICES THE PSEUDO-OP
;                                FCC.
;
;  INPUT.                        CHPTR POINTS TO CH.
;                                AFTER OP-CODE.
;
;  OUTPUT.                       WSPCE CLEARED.LENTH SET TO
;                                ZERO.LOCTR UPDATED.
;
;  CALLS.                        GCH51,GNC57,BIN15,
;                                ENE26,SCP23,DEC51.
;
;  CALLED-BY.                    SPR25.
;
;  GLOBAL-VARIABLES-USED.        PSFLG,WSPCE.
;
;  GLOBAL-VARIABLES-CHANGED.     CHPTR,LENTH,WSPC1,LOCTR.
;
;  ERROR-BITS-SET.               2(D),3
;
;
;  SPACE FOR SAVING ACC CONTENTS
;
AC055:   .BLK    1
AC155:   .BLK    1
AC255:   .BLK    1
AC355:   .BLK    1
;
;  CONSTANTS
;
CHP55:   .BLK    1
DEL55:   .BLK    1
COM55:           54                      ; COMMA
CHZ55:           60                      ; CH 9
CHN55:           71                      ; CH 9
CST55:           121
CHK55:           400                     ; DEC 256
```

```
TWO55:          2
THR55:          3
;
;  ENTRY SAVE ACCS.
;
FC155:  STA     0,AC055
        STA     1,AC155
        STA     2,AC255
        STA     3,AC355
;
; SET CHPTR
;
        JSRS    SCP23
        JMP     END55                   ; ERROR RETURN.
;
; CHPTR SET,SAVE VALUE,
; AND CHECK WHICH TYPE OF FNC.
;
        LDA     0,CHPTR
        STA     0,CHR55
        JSRS    GCH81
        STA     0,DEL55
;
; CHECK WHETHER CH IS DECIMAL
;
        LDA     1,CHZ55
        LDA     2,CHN55
        ADDZE   2,0,SNC
        ADDZE   0,1,SZC
        JMP     LCP55                   ; NOT DEC.
;
; CH. IS DECIMAL.
; EVALUATE AND CHECK IF BREAK IS #,*.
;
        DSZ     CHPTR
        JSRS    DEC31
;
; ACC 1 CONTAINS VALUE,ACC0 CONTAINS BREAK.
; CHECK WHETHER BREAK IS #,*.
;
        LDA     2,COM55
        SUBE    0,2,SZR
        JMP     LCP55                   ; NOT COMMA.
;
; VALUE IS IN ACC 1.
; CHECK WHETHER VALUE IS LESS THAN 256.
;
```

```
          LDA     0,CHK55
          SUBLE   0,1,SNC
          JMP     VGT55
;
;  GET CHARACTER AND PUNCH IT OUT
;
          NEG     1,1
LBL55:    IOV     1,1,SNR
          JMP     END55
          JSRS    GNC87
          MOVS    0,0
          STAS    0,WSPCE
          ISZ     LENTH
          JSRS    BIN15
          SUBO    0,0
          STAS    0,WSPCE
          STA     0,LENTH
          LDA     0,LOCTR
          INC     0,0
          STA     0,LOCTR
          INC     1,1
          JMP     LBL55
;
;  DELIMITER IS IN DEL55,CHPTR VALUE IS IN CHP55.
;  PUNCH IT OUT.
;
LOP55:    LDA     1,DEL55
          LDA     0,CHP55
          STA     0,CHPTR
;
;
;
NXT55:    JSRS    GNC87
          LDA     3,CHPTR
          LDA     2,CST55
          SUBE    3,2,SNR
          JMP     ERR55
;
;  CHECK IF DELIMITER
;
          SUBE    0,1,SNR
          JMP     END55
;
;  CHARACTER IS NOT DELIMITER,PUNCH IT OUT
;
          IOVS    0,0
          STAS    0,WSPCE
```

```
        ISZ     LENTH
        JSRS    BIN15
;
; SET LENTH AND WSPCE TO 0
;
        SUB     0,0
        STAS    0,WSPCE
        STA     0,LENTH
        LDA     0,LOCTR
        INC     0,0
        STA     0,LOCTR
        JMP     NXT55
;
; SET BIT NO. 3.
;
VGT55:  LDA     0,THR55
        STA     0,AG155
        JSRS    ENE26
AG155:  .BLK    1
        JMP     END55
;
; ERROR .SET BIT NO 2
;
ERR55:  LDA     0,TWO55
        STA     0,ARG55
        JSRS    ENE26
ARG55:  .BLK    1
;
; RESTORE ACC AND RETURN
;
END55:  LDA     0,AC055
        LDA     1,AC155
        LDA     2,AC255
        JMPS    AC355
;
;
;
; ***********************************************************
;
; NAME.                         FCB54.
;
; FUNCTION.                     THIS ROUTINE SERVICES THE
;                               PSEUDO-OP FCB.
;
; INPUT.                        CHPTR POINTS TO CH.
;                               AFTER OP-CODE.
;
```

```
;  OUTPUT.                          CHPTR POINTS TO BREAK.
;                                   LENTH ,WSPCE SET TO 0.
;  CALLS.                           ST304,EVL97,BIN15.
;  CALLED-BY.                       SPR25.
;  GLOBAL-VARIABLES-USED.           WSPCE,PSTLG.
;  GLOBAL-VARIABLES-CHANGED.        LENTH,WSPC1,LOCTR.
;  ERROR-BITS-SET.                  NONE.
;
;  SPACE FOR SAVING ACC CONTENTS.
;
AC054:    .BLK    1
AC154:    .BLK    1
AC254:    .BLK    1
AC354:    .BLK    1
;
;  CONSTANTS
;
ONE54:            1
COM54:            54
BRK54:    .BLK    1
;
;  ENTRY,SAVE ACC CONTENTS
;
FC154:    STA     0,AC054
          STA     1,AC154
          STA     2,AC254
          STA     3,AC354
;
;  SET CHPTR
;
          JSRS    ST304
          JMP     END54
;
;  EVALUATE EXPRESSION.
;
LBL54:    JSRS    EVL97
;
;  EXP HAS BEEN    EVALUATED.STORE IN WSPCE
;
          LDA     0,BREAK
```

```
        STA     0,BRK54
        MOVS    1,1
        STAS    1,WSPCE

;INCREMENT LENTH,LOCTR

        ISZ     LENTH

;CALL ON ROUTINE TO PUNCH IT OUT

        JSRS    BIN15

;SET LENTH TO 0,CLEAR WSPCE

        SUB     1,1
        STA     1,LENTH
        STAS    1,WSPCE
        LDA     1,LOCTR
        INC     1,1
        STA     1,LOCTR

;CHECK ON BREAK CHARACTER.

        LDA     0,BRK54
        LDA     1,COM54
        SUBE    0,1,SZR
        JMP     END54

;CHARACTER IS A COMMA,EVALUATE NEXT EXP

        ISZ     CHPTR
        JMP     LBL54

;RESTORE ACC CONTENTS AND RETURN

END54:  LDA     0,AC054
        LDA     1,AC154
        LDA     2,AC254
        JMPS    AC354

;*****************************************************

;NAME.                   FOR56.
;FUNCTION.               THIS ROUTINE SERVICES THE
```

```
;                                   PSEUDO-OP FDB.
; INPUT.
;                                   CHPTR POINTS TO CH.
;                                   AFTER OP-CODE.
; OUTPUT.
;                                   LENTH ,WSPCE SET TO 0.
;                                   CHPTR POINTS TO BREAK.
; CALLS.
;                                   ST664,EVL97,BIN15.
; CALLED-BY.
;                                   SPM25.
; GLOBAL-VARIABLES-USED.
;                                   WSPCE,PSFLG.
; GLOBAL-VARIABLES-CHANGED.
;                                   LENTH,WSPC1,LOCTR.
; ERROR-BITS-SET.
;                                   NONE.
;
;
; SPACE FOR SAVING ACC CONTENTS
;
AC056:   .BLK     1
AC156:   .BLK     1
AC256:   .BLK     1
AC356:   .BLK     1
;
; CONSTANTS
;
ONE56:            1
CON56:            54
BRK56:   .BLK     1
;
; ENTRY,SAVE ACCS.
;
FD156:   STA      0,AC056
         STA      1,AC156
         STA      2,AC256
         STA      3,AC356
;
; SET CHPTR
;
         JSR@     ST664
         JMP      END56
;
; EVALUATE EXPRESSION.
;
LBL56:   JSR@     EVL97
```

```
;
;  EXP HAS BEEN    EVALUATED.STORE IN WSPCE
;
        LDA     0,BREAK
        STA     0,BRK56
        STA$    1,WSPCE
;
;INCREMENT LENTH,SET LOCTR
;
        ISZ     LENTH
        ISZ     LENTH
;
;CALL ON ROUTINE TO PUNCH IT OUT
;
        JSR$    BI,15
;
;SET LENTH TO 0,CLEAR WSPCE
;
        SUB     1,1
        STA     1,LENTH
        STA$    1,WSPCE
        LDA     1,LOCTR
        INC     1,1
        INC     1,1
        STA     1,LOCTR
;
;CHECK ON BREAK CHARACTER.
;
        LDA     0,BRK56
        LDA     1,CO456
        SUBE    0,1,SZR
        JMP     END56
;
;CHARACTER IS A COMMA,EVALUATE NEXT EXP
;
        ISZ     CHPTR
        JMP     LBL56
;
;RESTORE ACC CONTENTS AND RETURN
;
END56:  LDA     0,AC056
        LDA     1,AC156
        LDA     2,AC256
        JMP$    AC356
;
;*****************************************************
```

```
;
; NAME.                          BOC46.
; FUNCTION.                      CONVERTS A BINARY NO.
;                                INTO A SERIES OF
;                                ASCII DECIMAL CHARACTERS.
; INPUT.                         BINARY NO. IN ACC 0.
; OUTPUT.                        ASCII CH. ARE INSERTED
;                                INTO PATR STARTING FROM
;                                PNTR.
; CALLS.                         EPT33.
; CALLED-BY.                     PAG66,ILC23,LER17.
; GLOBAL-VARIABLES-USED.         NONE.
; GLOBAL-VARIABLES-CHANGED.      PNTR.
; ERROR-BITS-SET.                NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC046:    .BLK    1
AC146:    .BLK    1
AC246:    .BLK    1
AC346:    .BLK    1
;
; POWERS OF TEN
;
          .RDX    10
PTN46:              10000
                    1000
                    100
                    10
                    1
                    0
          .RDX    8
PTA46:              PTN46
TEM46:    .BLK    1
ZER46:              60
;
;SAVE ACCS,ENTRY
```

```
;
BC146:  STA     0,AC046
        STA     1,AC146
        STA     2,AC246
        STA     3,AC346
;
; SET UP ADDRESS IN TEN46
;
        LDA     3,PTA46
        STA     3,TEN46
        MOV     0,2
;
; GET NEXT DIGIT AND GIVE IT OUT
;
LOP46:  LDAS    1,TEN46
        MOV     1,1,SNR
        JMP     END46
;
;
;
        LDA     0,ZER46
;
; ACC 2 CONTAINS RUNNING VALUE,ACC 0 CONTAINS COUNT
;
LEL46:  SUBZ    1,2,SZC
        INC     0,0,SKP
        ADC     1,2,SKP
        JMP     LEL46
;
; CH. IS IN ACC 0,GIVE IT OUT
;
        JSRS    EPT53
        ISZ     PRPT2
        DSZ     TE146
        JMP     LOP46
;
; RESTORE ACC AND RETURN
;
END46:  LDA     0,AC046
        LDA     1,AC146
        LDA     2,AC246
        JMPS    AC346
;
;
; ***************************************************
;
; NAME.
```

```
;
;  FUNCTION.                        NA157.
;
;                                   SERVICES THE PSEUDO-OP
;                                   NAM.GETS THE FIRST SIX CH.
;                                   AND PUTS THEM INTO NAME.
;  INPUT.
;                                   CHPTR POINTS TO CH.
;                                   AFTER OP-CODE.
;  OUTPUT.
;                                   NONE.
;  CALLS.
;                                   SCP23,GNC57.
;  CALLED-BY.
;                                   SPN25.
;  GLOBAL-VARIABLES-USED.
;                                   NAME.
;  GLOBAL-VARIABLES-CHANGED.
;                                   LCFLG,NAME1.
;  ERROR-BITS-SET.
;                                   NONE.
;
;  SPACE FOR SAVING ACCS.
;
AC057:   .BLK    1
AC157:   .BLK    1
AC257:   .BLK    1
AC357:   .BLK    1
;
;  CONSTANTS
;
TMP57:           3
CTR57:   .BLK    1
;
;  ENTRY POINT,SAVE ACCS.
;
NA157:   STA     0,AC057
         STA     1,AC157
         STA     2,AC257
         STA     3,AC357
         SUB     0,J
         STA     0,LCFLG
;
;  SET CHPTR AND ENTER
;
         JSRS    SCP23
         JMP     END57
```

```
;
          LDA     0,TMR57
          STA     0,CTR57
          DSZ     CNPTR
          LDA     2,NAME
;
; ENTER INTO NAME
;
LBL57:    JSR@    GNC87
          MOVS    0,1
          JSR@    GNC87
          ADD     0,1
          STA     1,0,2
          INC     2,2
          DSZ     CTR57
          JMP     LBL57
;
; RESTORE ACCS AND RETURN
;
END57:    LDA     0,AC057
          LDA     1,AC157
          LDA     2,AC257
          JMP@    AC357
;
;*****************************************************
;
; NAME.                         ENS31.
; FUNCTION.                     ENTERS NSPCE VALUE AFTER
;                               CONVERSION INTO HEX INTO
;                               LOCATIONS 16-24 OF PRTLN.
; INPUT.                        NONE.
; OUTPUT.                       NONE.
; CALLS.                        CVT84,BHX47,EPT03.
; CALLED-BY.                    PSC16.
; GLOBAL-VARIABLES-USED.        LENTH,NSPCE,VLFLG,VALUE.
; GLOBAL-VARIABLES-CHANGED.     PRPTR.
; ERROR-BITS-SET.
```

```
;                                       NONE.
;
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC031:    .BLK    1
AC131:    .BLK    1
AC231:    .BLK    1
AC331:    .BLK    1
;
; CONSTANTS.
;
CN031:            22                        ; DEC 18
CN131:            27
MSK31:            377
ONE31:            1
;
; SAVE ACC CONTENTS ,ENTRY POINT.
;
EW131:    STA     0,AC031
          STA     1,AC131
          STA     2,AC231
          STA     3,AC331
;
; SET UP PNTR VALUE
;
          LDA     0,CN031
          STA     0,PNTR
;
; CHECK VALUE OF LENTH
;
          LDA     0,LENTH
          MOV     0,0,SNR
          JMP     LZR31                     ; LENTH IS 0
;
; LENTH IS NOT ZERO,SET UP
; ADDRESS IN ACC 2,DETERMINE LENTH VALUE.
;
          LDA     2,NSPCE
          LDA     1,ONE31
;
          SUB     1,0,SNR
          JMP     LCN31                     ; LENTH IS 1
;
          SUB     1,0,SNR
          JMP     LTW31                     ; LENTH IS 2
;
```

```
;
; LENTH IS 3.
; ACC 2 CONTAINS ADDRESS OF BYTES.
; PRINT OUT FIRST TWO BYTES.
;
;
          LDA     0,J,2
          JSR@    CVT64
          LDA     0,CN131
          STA     0,PRPTR
          INC     2,2
;
; PUT OUT BYTE  1 OR 3
;
LCN31:    LDA     0,J,2
          JSR@    BHX47
AG131:    .BLK    1
AG231:    .BLK    1
;
; MOST SIGNIFICANT HALF IS ZERO.
;
          LDA     0,AG131
          MOVO    0,J
          LDA     1,MSK31
          ANDC    1,0
          JSR@    EPT63
          ISZ     PRPTR
;
          LDA     0,AG131
          AND     1,J
          JSR@    EPT63
          JMP     END31
;
; LENTH IS TWO.
;
LTW31:    LDA     0,J,2
          JSR@    CVT64
          JMP     END31
;
; LENTH IS 0,CHECK VALUE FLAG TO SEE IF
; VALUE HAS TO BE PRINTED.
;
LZR31:    LDA     0,VLFLG
          MOV     0,J,SNR
          JMP     END31
;
; VALUE FLAG IF NON ZERO.
;
```

```
          LDA     0,VALUE
          JSR@    CVT64

;  RESTORE ACCS AND RETURN.

END31:    LDA     0,AC031
          LDA     1,AC131
          LDA     2,AC231
          JMP@    AC331

;
;  ********************************************************

;  NAME.                          PED20.
;  FUNCTION.                      PUNCHES E-O-F RECORD.
;  INPUT.                         NONE.
;  OUTPUT.                        NONE.
;  CALLS.                         PNL12,NRT36.
;  CALLED-BY.                     NOUP3.
;  GLOBAL-VARIABLES-USED.         BMOV,BNFLG.
;  GLOBAL-VARIABLES-CHANGED.      OUTOV.
;  ERROR-BITS-SET.                NONE.

;  SPACE FOR SAVING ACC CONTENTS

AC320:    .BLK    1
AC020:    .BLK    1

;  CONSTANTS

EOF20:              34523
                    31460
                    30060
                    30060
                    41506
                    0
EOA20:              EOF20
```

```
;
; ENTRY,SAVE ACCS.
;
PE120:  STA     0,AC020
        STA     3,AC320
;
; CHECK BNFLG TO SEE IF NECC.
;
        LDA     0,BNFLG
        IOV     0,0,SNR
        JMP     END20
;
; PUNCH OUT NULLS.
;
        JSRS    PNL12
;
; PUNCH OUT RECORD
;
        LDA     0,BPNOV
        STA     0,OUTOV
        LDA     2,EOA20
        JSRS    WRT66
;
; RESTORE ACCS AND RETURN
;
END20:  LDA     0,AC020
        JMPS    AC320
;
;*****************************************************
;
; NAME.                         SPC62.
; FUNCTION.                     SERVICE THE PSEUDO-OP
;                               SPC.
; INPUT.                        CHPTR POINTS TO CH.
;                               BEFORE OP-CODE.
; OUTPUT.                       CHPTR POINTS TO BREAK.
; CALLS.                        PLF06,ST804,EVL07.
; CALLED-BY.                    SPM25.
; GLOBAL-VARIABLES-USED.        NONE.
```

```
;GLOBAL-VARIABLES-CHANGED.
;                                  SCFLG.
;ERROR-BITS-SET.                   NONE.
;
;
;SPACE FOR SAVING ACC CONTENTS.
;
AC062:   .BLK    1
AC162:   .BLK    1
AC262:   .BLK    1
AC362:   .BLK    1
;
;SAVE ACCS,ENTRY
;
SP162:   STA     0,AC062
         STA     1,AC162
         STA     2,AC262
         STA     3,AC362
;
;EVALUATE AFTER SETTING CHPTR
;
         JSRS    ST304
         JMP     END62
;
;EVALUATE EXP.NUMBER IS INTERPRETED AS A BIN NO.
;
         JSRS    EVL97
         MOV     0,0,SNR
         JMP     END62
;
;NO ERROR IN EVALUATION,CALL ON
;ROUTINE TO PRINT OUT LFDS.
;
         MOV     1,0
         JSRS    PLF03
;
;SET SCFLG .
;
         SUB     1,1
         STA     1,SCFLG
;
;RESTORE ACCS,RETURN.
;
END62:   LDA     0,AC062
         LDA     1,AC162
         LDA     2,AC262
```

```
        JMPS    AC362
;
; ****************************************************
;
; NAME.                          PN803.
; FUNCTION.                      PUNCHES OUT THE ≠START≠
;                                RECORD WITH 3 WORDS
;                                OF ≠NAME≠.
; INPUT.                         NONE.
; OUTPUT.                        NONE.
; CALLS.                         WRT36,PNL12,SHX47.
; CALLED-BY.                     INT13.
; GLOBAL-VARIABLES-USED.         BPNCV,SNFLG,NAME.
; GLOBAL-VARIABLES-CHANGED.      OUTCV.
; ERROR-BITS-SET.                NONE.
;
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC803:    .BLK     1
A1803:    .BLK     1
A2803:    .BLK     1
A3803:    .BLK     1
;
; CONSTANTS AND COUNTERS.
;
ST803:             30123               ;  S.
                   33060               ;  BYTE COUNT
                   30060               ; ADDRESS
                   30060               ; ADDRESS
NM803:    .BLK     3
CH803:    .BLK     1
                   0
NA803:               NM803
SA803:             ST803
CT803:             3
CN803:    .BLK     1
```

```
SM8C3:              113220
MS8C3:              377
NC8C3:    .BLK      1
LC8C3:    .BLK      1
;
; SAVE ACCS,ENTRY
;
P18C3:    STA       0,A0803
          STA       1,A1803
          STA       2,A2803
          STA       3,A3803
;
; CHECK IF PUNCHING IS NEED.
;
          LDA       0,BNFLG
          MOV       0,0,SNR
          JMP       EN8C3
;
; PUNCH OUT NULLS.
;
          JSRS      PNL12.
;
; SET UP DEVICE CODES.
;
          LDA       0,BPNDV
          STA       0,OUTDV
;
; SET UP ADDRESSES
;
          LDA       0,NAME
          STA       0,NC803
          LDA       0,NA803
          STA       0,LC8C3
          LDA       0,CT803
          STA       0,CN803
;
          LDA       1,MS803
          LDA       2,SM803
;
; GET NAME,STORE IT,UPDATE SUM
;
LBP03:    LDAS      0,NC803
          MOVS      0,0
          STAS      0,LC803
;
          AND       1,0
```

```
            ADD     0,2
            LDA@    0,NC803
            AND     1,0
            ADD     0,2
;
            ISZ     NC803
            ISZ     LC803
            DSZ     CN803
            JMP     LB803
;
; ALL WORDS ENTERED,GET CHECKSUM
;
            COM     2,2
            MOV     2,0
            JSR@    BFX47
G1803:      .BLK    1
G2803:      .BLK    1
            LDA     2,G2803
            MOVS    2,2
            STA     2,CH803
;
; PUNCH OUT
;
            LDA     2,SA803
            JSR@    WRT86
;
; RESTORE ACCS AND RETURN
;
EN803:      LDA     0,A0803
            LDA     1,A1803
            LDA     2,A2803
            JMP@    A3803
; **********************************************************
;
; NAME.                         PAG60.
;
; FUNCTION.                     SERVICES THE PSEUDO-OP PAG.
;                               PRINTS OUT HEADING AFTER
;                               UPDATING PAGNO.
;
; INPUT.                        NONE.
;
; OUTPUT.                       NONE.
;
; CALLS.                        PLF08,PRT52,BDC46,WRT86.
;
; CALLED-BY.
```

```
;
; GLOBAL VARIABLES USED.          SPN25,PVL34.
;                                 LTFLG,SLOTD,PRTLN,NAME.
;                                 ENDFG.
; GLOBAL-VARIABLES-CHANGED.
;                                 OUTDV,PRTL1,SCFLG,PRPTR.
; ERROR-BITS-SET.
;                                 NONE.
;
;
; SPACE FOR SAVING ACC CONTENTS.
;
AC060:  .BLK    1
AC160:  .BLK    1
AC260:  .BLK    1
AC360:  .BLK    1
;
; CONSTANTS,COUNTERS
;
MES60:  .TXT    - ER. LINE. LOC.  VALUE.      INPUT.-
MEA60:          MES60
BLK60:          20040
CNT60:          74
NBK60:          0
NLE60:          4
NLA60:          4
NCH60:          3
SPN60:          74                      ; DEC 60.
PVL60:          102                     ; DEC 66.
PGE60:          120                     ; P
                101                     ; A
                107                     ; G
                105                     ; E
PGA60:          PGE60
CST60:          4
CTF60:  .BLK    1
;
; ENTRY,SAVE ACCS.
;
PA160:  STA     0,AC060
        STA     1,AC160
        STA     2,AC260
        STA     3,AC360
;
; CHECK WHETHER RECD.
;
        LDA     0,LTFLG
```

```
            NOV     0,0,SNR
            JMP     END60
            LDA     0,ENDFG
            NOV     0,0,SZR
            JMP     END60
;
;   PRINT OUT LEDS.
;
            LDA     0,NL360
            JSRS    PLF05
;
;   SET OUTPUT DEVICE CODE
;
            LDA     0,SLOTD
            STA     0,OUTDV
;
;   CLEAR PRINT LINE
;
            LDA     1,CNT60
            NEG     1,1
            LDA     2,PRTLN
            LDA     0,BLK60
LBL60:      STA     0,0,2
            INC     2,2
            INC     1,1,SZR
            JMP     LBL60
;
;   INTRODUCE NAME
;
            LDA     2,PRTLN
            LDA     1,NBK60
            ADD     1,2
            LDA     3,NAME
            LDA     1,NCH60
            NEG     1,1
            LDA     0,0,3
            STA     0,0,2
            INC     3,3
            INC     2,2
            INC     1,1,SZR
            JMP     .-5
;
;   ENTER #PAGE#.
;
            LDA     0,CST60
            STA     0,CTR60
;
```

```
          LDA     0,SPN60
          STA     0,PRPTR
;
          LDA     2,PGA60
;
LOP60:    LDA     0,0,2
          JSR@    EPT63
          INC     2,2
          ISZ     PRPTR
          DSZ     CTR60
          JMP     LOP60
;
;;; ENTER PAGE NO.
;
          LDA     0,PAGNO
          INC     0,0
          STA     0,PAGNO
          LDA     1,PWL60
          STA     1,PRPTR
          JSR@    ADC46
;
;;; ALL WORDS ARE ENTERED,PRINT OUT
;
          JSR@    PRT62
;
          SUB     0,0
          INC     0,0
          JSR@    PLF03
          LDA     2,NEA60
          JSR@    WRT65
; PRINT OUT LINE-FEEDS.
;
          LDA     0,NLA60
          JSR@    PLF00
;
;;; SET FLAG SO THAT LINE IS NOT PRINTED OUT
;
          SUB     0,0
          STA     0,SOFLG
;
;;; RESTORE ACCS AND RETURN.
;
ENDGO:    LDA     0,AC060
          LDA     1,AC160
          LDA     2,AC260
          JMP@    AC360
; ************************************************************
```

```
;
; NAME.                              MES17.
; FUNCTION.                          LISTS OUT THE
;                                    TOTAL NO. OF ERRORS.
; INPUT.                             NONE.
; OUTPUT.                            NONE.
; CALLS.                             BOC46,PRTS2.
; CALLED-BY.                         PASS1,WCUP3.
; GLOBAL-VARIABLES-USED.             PRTLN,SLCTD,PSFLG,EF1TB,
;                                    E1EPT,EF2TB,E2EPT.
; GLOBAL-VARIABLES-CHANGED.          PRTL1,OUTCV,PRPTR.
; ERROR-BITS-SET.                    NONE.
;
;
; SPACE FOR SAVING ACCS.
AC017:    .BLK     1
AC117:    .BLK     1
AC217:    .BLK     1
AC317:    .BLK     1
;
; MESSAGE
;
MES17:             52117                    ; TO
                   52101                    ; TA
                   46040                    ; L
                   42522                    ; ER
                   51117                    ; RO
                   51123                    ; RS
                   20040                    ;≠  ≠.
MEA17:             MES17
CST17:             20
CNT17:             74
BLK17:             20040
ONE17:             1
NWD17:             7
;
; SAVE ACCS,ENTRY
```

```
;
ME117:  STA     0,AC017
        STA     1,AC117
        STA     2,AC217
        STA     3,AC317
;
;;; SET DEVICE CODES
;
        LDA     0,SLOTC
        STA     0,OUTSV
;
;;; CLEAR TABLE
;
        LDA     2,PRTLN
        LDA     0,BLK17
        LDA     1,CNT17
        NEG     1,1
;
        STA     0,0,2
        INC     2,2
        INC     1,1,SZR
        JMP     .-3
;
;;; STORE MESSAGE
;
        LDA     2,PRTLN
        LDA     3,MEA17
        LDA     1,NWD17
        NEG     1,1
        LDA     0,0,3
        STA     0,0,2
        INC     2,2
        INC     3,3
        INC     1,1,SZR
        JMP     .-5
;
;;; GET NO OF ERRORS CONVERT AND ENTER
;
        LDA     0,PSFLG
        LDA     1,ONE17
        SUBE    0,1,SNR
        JMP     PCN17
;
;;; PASS OS 2
;
        LDA     1,ER2TB
        LDA     0,E2EPT
```

```
        JMP     NXT17
;
; PASS IS 1
;
PON17:  LDA     1,ER1TB
        LDA     0,E1EPT
;
NXT17:  SUBOR    1,0
        LDA     1,CST17
        STA     1,PRPTR
        JSR$    3CC45
        JSR$     PRTS2
;
; RESTORE ACCS AND RETURN
;
        LDA     0,AC017
        LDA     1,AC117
        LDA     2,AC217
        JMP$    AC317
;
; ****************************************************
;
; NAME.                         PLFJ8.
; FUNCTION.                     PRINTS OUT CRT AND LFDS.
; INPUT.                        ACC 0.CONTAINS THE NO.
;                               OF LFDS TO BE PUT OUT.
; OUTPUT.                       NONE.
; CALLS.                        PUT75.
; CALLED-BY.                    LOTS.
; GLOBAL-VARIABLES-USED.        LTFLG,SLOTD.
; GLOBAL-VARIABLES-CHANGED.     OUTDV.
; ERROR-BITS-SET.               NONE.
;
; SPACE FOR SAVING ACC CONTENTS.
;
```

```
AC0C8:    .BLK    1
AC208:    .BLK    1
AC308:    .BLK    1
;
; CONSTANTS
;
LFC08:            12
CRT08:            15
;
; SAVE ACCS,ENTRY
;
PL108:    STA     0,AC008
          STA     2,AC208
          STA     3,AC308
;
; CHECK LTFLG
;
          LDA     0,LTFLG
          MOV     0,0,SNR
          JMP     END08
;
; SET DEVICE CODES.
;
          LDA     3,SLOTD
          STA     3,OUTDV
;
;   PRINT LF AND CR
;
          LDA     0,CRT08
          JSR@    PUT75
          LDA     0,LFC08
          LDA     2,AC108
          NEG     2,2
          JSR@    PUT75
          INC     2,2,SZR
          JMP     .-3
;
;  RETURN
;
END08:    LDA     0,AC008
          LDA     2,AC208
          JMP@    AC308
;
;
; ********************************************************
;
; NAME.
```

```
;
; FUNCTION.                     PNL12.
;                               PUNCHES OUT CR,LF AND NULLS
; INPUT.                        NONE.
; OUTPUT.                       NONE.
; CALLS.                        PUT75.
; CALLED-BY.                    PED20,PUN40,PNE05.
; GLOBAL-VARIABLES-USED.        BNFLG,BPNDV.
; GLOBAL-VARIABLES-CHANGED.     OUTDV.
; ERROR-BITS-SET.               NONE.
;
;
; SPACE FOR SAVING ACCS.
;
AC012:   .BLK     1
AC212:   .BLK     1
AC312:   .BLK     1
;
; CONSTANTS
;
CRT12:            15
LFD12:            12
NNL12:            6
;
; ENTRY,SAVE ACCS.
;
PNL12:   STA      0,AC012
         STA      2,AC212
         STA      3,AC312
;
; CHECK BNFLG
;
         LDA      0,BNFLG
         MOV      0,0,SNR
         JMP      END12
;
; SET OUTPUT DEVICE ADDRESS
;
         LDA      0,BPNDV
         STA      0,OUTDV
```

```
;
; PRINT OUT CR,LF
;
        LDA     0,CRT12
        JSR@    PUT75
        LDA     0,LFD12
        JSR@    PUT75
;
; PRINT OUT NULLS
;
        SUB     0,0
        LDA     2,NNL12
        NEG     2,2
        JSR@    PUT75
        INC     2,2,SZR
        JMP     .-2
;
; RESTORE ACCS AND RETURN
;
END12:  LDA     0,AC012
        LDA     2,AC212
        JMP@    AC312
;
;
; ***************************************************************
;
; NAME.                          FVR04.
; FUNCTION.                      DOES VERTICAL FORMATTING.
; INPUT.                         NONE.
; OUTPUT.                        NONE.
; CALLS.                         PAG60.
; CALLED-BY.                     PSC10.
; GLOBAL-VARIABLES-USED.         OPFTB,LNCTR.
; GLOBAL-VARIABLES-CHANGED.      NONE.
; ERROR-BITS-SET.                NONE.
;
; SPACE FOR SAVING ACC. CONTENTS.
```

```
;
AC034:    .BLK    1
AC134:    .BLK    1
AC234:    .BLK    1
AC334:    .BLK    1
;
; CONSTANTS,COUNTERS.
;
CTR34:    .BLK    1
CNT34:            24                    ; DEC 20.
ONE34:            1
;
; ENTRY,SAVE ACCS.
;
FV134:    STA     0,AC034
          STA     1,AC134
          STA     2,AC234
          STA     3,AC334
;
; CHECK IF FORMATTING IS NEED.
;
          LDA     3,OPFTB
          LDA     0,5,3
          LDA     1,ONE34
          SUBE    0,1,SNR
          JMP     END34
;
; CHECK FIRST TIME THRO.
;
          LDA     0,LNCTR
          LDA     1,ONE34
          SUBE    0,1,SZR
          JMP     NXT34
;
; FIRST TIME THRO,INITIALISE CTR34.
;
          LDA     0,CNT34
          STA     0,CTR34
;
; CHECK IF END OF PAGE.
;
NXT34:    DSZ     CTR34
          JMP     END34
;
; PRINT HEADING.
;
          JSRS    PAGEH
```

```
;
; RESET COUNTER.
;
          LDA      0,CNT34
          STA      0,CTR34
;
;
; RESTORE ACCS AND RETURN.
END34:    LDA      0,AC034
          LDA      1,AC134
          LDA      2,AC234
          JMPS     AC334
;
;
;
;
; ****************************************************************
;
;
; NAME.                              LSM18.
; FUNCTION.                          LISTS OUT THE SYMBOL TABLE.
; INPUT.                             NONE.
; OUTPUT.                            NONE.
; CALLS.                             PSY39,PLF08.
; CALLED-BY.                         NOUP3.
; GLOBAL-VARIABLES-USED.             LTFLG,SYMTB,OPFTB,SLOTD.
; GLOBAL-VARIABLES-CHANGED.          OUTDV.
;
;
;
; SPACE FOR SAVING ACC. CONTENTS.
;
AC018:    .BLK     1
AC118:    .BLK     1
AC218:    .BLK     1
AC318:    .BLK     1
;
; CONSTANTS,COUNTERS.
;
```

```
LEN18:  .BLK    1                       ; SYMBOL TABLE
                                         LENGTH.
;
CTR18:  .BLK    1
FOR18:          4
SET18:          100000
SYM18:  .BLK    1
FIN18:  .BLK    1
LAD18:          LSP18
LSP18:  .BLK    3
BLN18:          10
;
; SAVE ACCS. ENTRY.
;
LS118:  STA     0,AC018
        STA     1,AC118
        STA     2,AC218
        STA     3,AC318
;
; CHECK WHETHER LISTING IS NEED.
;
        LDA     0,LTFLG
        MOV     0,0,SNR
        JMP     END18
;
; SET DEVICE CODES.
;
        LDA     0,SLOTC
        STA     0,OUTDV
;
; PRINT OUT BLANK LINES.
;
        LDA     0,BLN18
        JSRS    PLF00
;
; SYMBOL TABLE LISTING IS NEED.
;
; A METHOD WHICH DOES NOT USE EXTRA STORAGE ,BUT
; USES A LOT OF REDUNDANT COMPARISONS HAS BEEN USED.
;
; SET LEN18 TO SYMBOL TABLE LENGTH.
;
        LDA     3,OPETB
        LDA     0,3,3
        STA     0,LEN18
;
```

```
; SET ALL ZERO ENTRIES TO 177777.
;
            LDA     0,LEN16
            STA     0,CTR16
;
            LDA     3,SYMTB
            LDA     2,FOR16
            LDA     1,SET15
;
LB116:      LDA     0,0,3
            MOV     0,0,SNR
            STA     1,0,3
;
            ADD     2,3
            DSZ     CTR16
            JMP     LB116
;
;
; ALL NULL ENTRIES HAVE BEEN SET TO 177777.
; DETERMINE MINIMUM ENTRY.
; GET OUT OF LOOP IF MINIMUM VALUE IS 177777.
;
LB216:      LDA     2,SYMTB
            LDA     3,LAD16
            LDA     0,LEN16
            STA     0,CTR16
            LDA     0,SET16
            STA     0,LSP16
            SUB     0,0
            STA     0,FIN16
;
; ALL VALUES HAVE BEEN INITIALISED.
; ACC 2 CONTAINS ADDRESS OF SYMBOL TABLE.
; ACC 3 CONTAINS ADDRESS OF LOCAL SPACE.
; GO INTO LOOP TO DETERMINE MINIMUM VALUE.
;
LB316:      LDA     0,0,2
            LDA     1,0,3
            SUBE    0,1,SNR
            JMP     ZR116
;
; RESULT IS NOT ZERO.COMPARE ENTRIES.
;
            SUBLE   0,1,SZC
            JMP     NXT16
```

```
          JMP     REP18
;
; FIRST WORD MATCHES. CHECK SECOND WORD.
;
ZR118:    LDA     0,1,2
          LDA     1,1,3
          SUBEZ   0,1,SNR
          JMP     ZR218
;
; COMPARE ENTRIES.
;
          SUBLE   0,1,SZC
          JMP     NXT18
          JMP     REP18
;
; FIRST TWO ENTRIES MATCH.
;
ZR218:    LDA     0,2,2
          LDA     1,2,3
          SUBLE   1,0,SNC
          JMP     NXT18
;
; REPLACE MINIMUM VALUE.
;
REP18:    STA     2,FIN18
          LDA     0,0,2
          STA     0,0,3
          LDA     0,1,2
          STA     0,1,3
          LDA     0,2,2
          STA     0,2,3
;
; MINIMUM VALUE REPLACED.
; UPDATE ACC 2,CHECK WHETHER ALL
; ENTRIES HAVE BEEN CHECKED.
;
NXT18:    LDA     0,FOR18
          ADC     0,2
          DSZ     CTR18
          JMP     LB318
;
; FIN18 CONTAINS ADDRESS OF MINIMUM VALUE.
; LSP18 CONTAINS MINIMUM VALUE.
; CHECK WHETHER FIRST WORD IS 100000
;
          LDA     0,LSP18
          LDA     1,SET18
```

```
            SUBZ    0,1,SNR
            JMP     OVR18
;
;   ALL ENTRIES HAVE NOT BEEN PRINTED OUT.
;   PRINT OUT THIS ENTRY AND SET FIRST BIT TO 1.
;   CALL ON ROUTINE PSM39 TO PRINT OUT THE ENTRY.
;   PSM39 NEEDS AS INPUT THE ADDRESS OF ENTRY IN ACC 2.
;
            LDA     2,FIN18
            JSRS    PSY39
;
;   SET BIT 0 TO 1.
;
            LDAS    0,FIN18
            LDA     1,SET18
            ADD     1,0
            STAS    0,FIN18
;
;   GET NEXT MINIMUM VALUE.
;
            JMP     LB218
;
;   ALL ENTRIES HAVE BEEN PRINTED OUT.
;   CLEAR BIT 0 OF ALL ENTRIES.
;   IE. SUBTRACT 100000 FROM ALL ENTRIES.
;
OVR18:      LDA     0,LEN18
            STA     0,CTR18
            LDA     3,SYMTB
            LDA     2,FOR18
            LDA     1,SET18
;
LB418:      LDA     0,0,3
            SUB     1,0
            STA     0,0,3
;
            ADD     2,3
            DSZ     CTR18
            JMP     LB418
;
;   RESTORE ACCS AND RETURN.
;
END18:      LDA     0,AC018
            LDA     1,AC118
            LDA     2,AC218
            JMPS    AC318
;
```

```
;*****************************************************

; NAME.                          PSY39
; FUNCTION.                      PRINTS OUT AN ENTRY OF
;                                THE SYMBOL TABLE.
; INPUT.                         AC2 CONTAINS THE STARTING
;                                ADDRESS OF ENTRY.
; OUTPUT.                        NONE.
; CALLS.                         PUT75,BHX47.
; CALLED-BY.                     LSM16.
; GLOBAL-VARIABLES-USED.         SYMTB.
; GLOBAL-VARIABLES-CHANGED.      NONE.

; SPACE FOR SAVING ACC. CONTENTS.

AC039:    .BLK    1
AC139:    .BLK    1
AC239:    .BLK    1
AC339:    .BLK    1

; CONSTANTS,COUNTERS.

BLK39:            40
MSK39:            177
CNT39:            3
CTR39:    .BLK    1
CRT39:            15
LFD39:            12
NBK39:            10                    ; NO. OF BLANKS
                                          ; NECESSARY.

; ENTRY POINT,SAVE ACC CONTENTS.

PS139:    STA     0,AC039
          STA     1,AC139
          STA     2,AC239
```

```
          STA     3,AC339
;
;   PRINT OUT ENTRY.
;   ACC 2 CONTAINS STARTING ADDRESS.
;
          LDA     0,CNT39
          STA     0,CTR39
          LDA     1,MSK39
;
;
LB139:    LDA     0,0,2
          MOVS    0,0
          ADC     1,0,SNR
          LDA     0,BLK39
          JSRS    PUT75
;
;   PRINT OUT SECOND CHARACTER.
;
          LDA     0,0,2
          AND     1,0,SNR
          LDA     0,BLK39
          JSRS    PUT75
;
;
          INC     2,2
          DSZ     CTR39
          JMP     LB133
;
;   ALL WORDS PRINTED OUT.
;   PRINT OUT BLANKS.
;
          LDA     0,BLK39
          LDA     1,MSK39
          STA     1,CTR39
          JSRS    PUT75
          DSZ     CTR39
          JMP     .-2
;
;   ALL BLANKS PRINTED OUT.
;   CONVERT LOC. VALUE INTO HEX. AND PRINT OUT.
;
          LDA     2,AC239
          LDA     0,3,2
          JSRS    3HX47
AC139:    .BLK    1
AC239:    .BLK    1
;
```

```
;
; PRINT OUT 4 CHARACTERS.
;
        LDA     0,AG139
        MOVS    0,0
        JSR@    PUT75
        MOVS    0,0
        JSR@    PUT75
;
        LDA     0,AG239
        MOVS    0,0
        JSR@    PUT75
        MOVS    0,0
        JSR@    PUT75
;
; PRINT OUT CR,LF.
;
        LDA     0,CRT39
        JSR@    PUT75
        LDA     0,LFD39
        JSR@    PUT75
;
; ALL WORDS PRINTED OUT.
; RESTORE ACCS. AND RETURN.
;
END39:  LDA     0,AC039
        LDA     1,AC139
        LDA     2,AC239
        JMP@    AC339
;
;*****************************************************
;
;
        .LOC    12000
;
;*****************************************************
;
;           TABLES (ARRAYS).
;
;*****************************************************
;
```

```
; SYMBOL TABLE.
SYMT1:  .BLK    2000

        .BOX    10

; OBJECT-CODE BUFFER.
OBCU1:  .BLK    40

        .TXT*   1

; DEFAULT-NAME.
DEFN1:  .TXT    /START /

; DIRECT ADDRESS TABLE.
DIRA1:  .BLK    50

; ERROR TABLE FOR PASS-1.
ERT1:   .BLK    50

; ERROR-TABLE FOR PASS-2.
ERT2:   .BLK    50

; INPUT SOURCE-LINE.
INBL1:  .BLK    40

; ********************************************************

; MACHINE-OPERATION TABLE.
```

```
!OPT1!   .TXT    .AJA.
                  2075
         .TXT    /ADC/
                   157
         .TXT    /ADD/
                   139
         .TXT    /AND/
                   132
         .TXT    /ASL/
                   534
         .TXT    /ASR/
                   533
         .TXT    /BCC/
                  1828
         .TXT    /BCS/
                  1838
         .TXT    /BEQ/
                  1831
         .TXT    /BGE/
                  1835
         .TXT    /BGT/
                  1833
         .TXT    /BHI/
                  1826
         .TXT    /BIT/
                   133
         .TXT    /BLE/
                  1839
         .TXT    /BLS/
                  1827
         .TXT    /BLT/
                  1837
         .TXT    /BMI/
                  1835
         .TXT    /BNE/
                  1830
         .TXT    /BPL/
                  1834
         .TXT    /BRA/
                  1824
         .TXT    /BSR/
                  1833
         .TXT    /BVC/
                  1832
         .TXT    /BVS/
                  1833
         .TXT    /CBA/
```

```
.TXT    /CMA/
          138
.TXT    /PSH/
          822
.TXT    /PUL/
          816
.TXT    /ROL/
          585
.TXT    /ROR/
          582
.TXT    /RTI/
         2107
.TXT    /RTS/
         2105
.TXT    /SBA/
         2064
.TXT    /SBC/
          130
.TXT    /SEC/
         2061
.TXT    /SEI/
         2063
.TXT    /SEV/
         2059
.TXT    /STA/
          407
.TXT    /STS/
         1439
.TXT    /STX/
         1503
.TXT    /SUB/
          120
.TXT    /SWI/
         2111
.TXT    /TAB/
         2076
.TXT    /TAP/
         2054
.TXT    /TBA/
         2071
.TXT    /TPA/
         2055
.TXT    /TST/
          589
.TXT    /TSX/
         2036
.TXT    /TXS/
```

```
        .TXT    2101
                /WAI/
                2110
;
; ************************************************************
;
; NAME(CURRENT).
NAME1:  .BLK    3
;
; OPTIONS DEFAULT TABLE.
OPDT1:          2
                 1
                2
                255
                2
                2
;
; OPTION-FLAG TABLE.
OPFT1:  .BLK    6
;
; ************************************************************
;
; PSEUDO-OP TABLE.
;
PОPT1:  .TXT     /END/
                EN152
        .TXT     /EQU/
                EQ153
        .TXT     /FCB/
                FC154
        .TXT     /FCC/
                FC155
        .TXT     /FDB/
                FC156
        .TXT     /NAM/
                NA157
        .TXT     /OPT/
```

```
                OP158
        .TXT      /ORG/
                OR159
        .TXT    /PAG/
                PA160
        .TXT    .RB.
                RM161
        .TXT      /SPC/
                SP162

;*****************************************************

; PRINT LINE.

PRTL1:  .BLK    60

; WORKING SPACE.SPACE FOR
; ASSEMBLING INSTRUCTION.

WSPC1:  .BLK    2

        .RDX    8

        .TXTM   0

;*****************************************************

; STUBS FOR OTHER ROUTINES.

OP158:  JMP     0,3

;*****************************************************

        .END
```

APPENDIX I

SAMPLE PROGRAMS

SAMPLE PROGRAM #1

Exhaustive testing of

Various Instruction Types

and Addressing Modes

```
START                                                    PAGE  00001
 ER. LINE. LOC.  VALUE.  INPUT

     00001              **
     00002              * TEST INSTRUCTIONS
     00003              **
     00004              *
     00005              *
     00006        0010  AAA     EQU     $10
     00007        0110  BBB     EQU     $110
     00008        FFAB  CCC     EQU     $FFAB
     00009        0300  DDD     EQU     $300
     00010              *
     00011              *
     00012 0100                 ORG     $100
     00013              *
     00014              * FORMAT NO. 0
     00015              **
     00016              *
     00017              * IMMEDIATE
     00018              **
     00019 0100  8190           ADC A   #$10
     00020 0102  C910           ADC B   #AAA
```

```
 ER. LINE. LOC.  VALUE.          INPUT.

     00021 0104  8920                    ADC A   #AAA+$10
     00022 0106  C931                    ADC B   #'1
     00023               **/
     00024               * DIRECT
     00025               **
     00026 0108  9910                    ADC A   $10
     00027 010A  D910                    ADC B   AAA
     00028 010C  990B                    ADC A   AAA-$5
     00029               *
     00030               * EXTENDED
     00031               *
     00032 010E  F901 00                 ADC B   $100
     00033 0111  B9FF AB                 ADC A   CCC
     00034 0114  F9FF 00                 ADC B   CCC-$AB
     00035               *
     00036               * INDEXED
     00037               *
     00038 0117  A900                    ADC A   X
     00039 0119  E900                    ADC B   ,X
     00040 011B  A910                    ADC A   $10,X
```

START PAGE 00003

```
ER. LINE. LOC.  VALUE.          INPUT.

    00041 011D  E910                    ADC B   AAA,X
    00042 011F  A920                    ADC A   AAA*$2,X
    00043               *
    00044               *
    00045               * FORMAT NO. 1
    00046               *
    00047               *
    00048               * DIRECT
    00049               *
    00050 0121  9710                    STA A   $10
    00051 0123  D710                    STA B   AAA
    00052 0125  970B                    STA A   AAA-$5
    00053               *
    00054               * EXTENDED
    00055               *
    00056 0127  F701 00                 STA B   $100
    00057 012A  B7FF AB                 STA A   CCC
    00058 012D  F7FF 00                 STA B   CCC-$AB
    00059               *
    00060               * INDEXED
```

```
 ER. LINE. LOC.  VALUE.     INPUT.

     00061              *
     00062 0130  A700             STA A          X
     00063 0132  E700             STA B          ,X
     00064 0134  A710             STA A          $10,X
     00065 0136  E710             STA B          AAA,X
     00066 0138  A708             STA A          AAA/$2,X
     00067              *
     00068              *
     00069              * FORMAT NO. 2
     00070              *
     00071              *
     00072              * ACC A,B
     00073              *
     00074 013A  48               ASL A
     00075 013B  58               ASL B
     00076              *
     00077              * EXTENDED
     00078              *
     00079 013C  7801 00          ASL            $100
     00080 013F  78FF AB          ASL            CCC
```

START PAGE 00005

```
 ER. LINE. LOC.  VALUE.          INPUT.

     00081 0142  78FF 00                ASL   CCC-$AB
     00082                      *
     00083                      * INDEXED
     00084                      *
     00085 0145  6800                   ASL   X
     00086 0147  6800                   ASL   ,X
     00087 0149  6810                   ASL   $10,X
     00088 014B  6810                   ASL   AAA,X
     00089 014D  6812                   ASL   AAA+$2,X
     00090                      **
     00091                      * FORMAT NO. 3
     00092                      **
     00093                      *
     00094                      * ACC A,B
     00095                      *
     00096 014F  36                     PSH A
     00097 0150  37                     PSH B
     00098                      *
     00099                      * FORMAT NO. 4
     00100                      *
```

START PAGE 00006

```
 ER. LINE. LOC.  VALUE.          INPUT.

     00101                  *
     00102                  * IMMEDIATE
     00103                  *
     00104 0151  8C01  00           CPX       #$100
     00105 0154  8CFF  AB           CPX       #CCC
     00106 0157  8C00  10           CPX       #AAA
     00107 015A  8CFF  00           CPX       #CCC-$AB
     00108 015D  8C31  20           CPX       #'1
     00109                  *
     00110                  * DIRECT
     00111                  *
     00112 0160  9C10               CPX       $10
     00113 0162  9C10               CPX       AAA
     00114 0164  9C12               CPX       AAA+$2
     00115                  *
     00116                  * EXTENDED
     00117                  *
     00118 0166  BC01 00            CPX       $100
     00119 0169  BCFF AB            CPX       CCC
     00120 016C  BCFF 00            CPX       CCC-$AB
```

START PAGE 00007

```
 ER. LINE. LOC.  VALUE.          INPUT.

     00121              *
     00122              *
     00123              * INDEXED
     00124              *
     00125 016F  AC00           CPX     X
     00126 0171  AC00           CPX     ,X
     00127 0173  AC10           CPX     $10,X
     00128 0175  AC10           CPX     AAA,X
     00129 0177  AC12           CPX     AAA+$2,X
     00130              *
     00131              * FORMAT NO. 5
     00132              *
     00133              *
     00134              * DIRECT
     00135              *
     00136 0179  9F10           STS     $10
     00137 017B  9F10           STS     AAA
     00138 017D  9F12           STS     AAA+$2
     00139              *
     00140              * EXTENDED
```

START PAGE 00008

```
 ER. LINE. LOC.  VALUE.              INPUT.

     00141                *
     00142 017F  BF01 00                    STS   $100
     00143 0182  BFFF AB                    STS   CCC
     00144 0185  BFFF 00                    STS   CCC-$AB
     00145                *
     00146                *** INDEXED
     00147                *
     00148 0188  AF00                       STS   X
     00149 018A  AF00                       STS   ,X
     00150 018C  AF10                       STS   $10,X
     00151 018E  AF10                       STS   AAA,X
     00152 0190  AF12                       STS   AAA+$2,X
     00153                *
     00154                * FORMAT NO, 6
     00155                *
     00156                *
     00157                * EXTENDED
     00158                *
     00159 0192  7E01  00                   JMP   $100
     00160 0195  7EFF  AB                   JMP   CCC
```

START PAGE 00009

```
 ER. LINE. LOC.  VALUE.                  INPUT.

     00161 0198  7EFF  00                      JMP   CCC-$AB
     00162                   *
     00163                   *
     00164                   * INDEXED
     00165                   *
     00166 019B  6E00                          JMP   X
     00167 019D  6E00                          JMP   ,X
     00168 0197  6E10                          JMP   $10,X
     00169 01A1  6E10                          JMP   AAA,X
     00170 01A3  6E12                          JMP   AAA+$2,X
     00171                   *
     00172                   * FORMAT NO. 7
     00173                   *
     00174 0300                        ORG     $300
     00175 0300  2404                  BCC     $306
     00176 0302  24FC                  BCC     DDD
     00177 0304  240A                  BCC     DDD+$10
     00178                   *
     00179                   * FORMAT NO. 8
     00180                   *
```

START PAGE 00010

```
 ER. LINE. LOC.  VALUE.              INPUT.

     00181 0306  01                          NOP
     00182 0307  19                          DAA
     00183                   *
     00184                   * OTHER
     00185                   *
     00186 0308  8910                        ADCA    #AAA
     00187 030A  D910                        ADCB    AAA
     00188 030C  36                          PSHA
     00189 030D  58                          ASLB
     00190                   *
     00191 030E  9903                        ADC A   %11
     00192 0310  9909                        ADC A   @11
     00193 0312  9911                        ADC A   $11
     00194 0314  990B                        ADC A   11
     00195                   *
     00196 0316  9905                        ADC A   %11+%10
     00197 0318  9901                        ADC A   %11/%11
     00198 031A  9901                        ADC A   %11-%10
     00199 031C  9909                        ADC A   %11*%11
     00200                   *
```

```
START                                                    PAGE  00011
 ER. LINE. LOC.  VALUE.          INPUT.

     00201 0200                          ORG    $200
     00202                    *
     00203                    *
     00204 0200                          FCB    $FF,10,,%11
     00205 0204                          FDB    ,$F,$FFFF,,%11
     00206                    *
     00207                    *
     00208 020E                          FCC    /TEXT/
     00209 0212                          FCC    6,TEXT
     00210 0218  0010                    RMB    $10
     00211 0228  16                      TAB
     00212                               END
TOTAL ERRORS   00000


AAA             0010
BBB             0110
CCC             FFAB
DDD             0300
```

SAMPLE PROGRAM #2

Hexa-Decimal

Memory Dump

Program

DMP PAGE 00001

```
 ER. LINE. LOC.  VALUE.   INPUT.

     00001               NAM     DMP
     00002           **
     00003           *   ALTAIR 680B HEXADECIMAL MEMORY
                         DUMP PROGRAM.
     00004           **
     00005           *   LOAD VIA PROM MONITOR.
     00006           **
     00007           *   USE MONITOR J COMMAND TO
     00008           *   START EXECUTION AT 0005.
     00009           **
     00010           *   ENTER ADDRESS OF FIRST BYTE TO DUMP.
     00011           **
     00012           *   ENTER ADDRESS OF LAST BYTE TO DUMP.
     00013           **
     00014           *   TYPE ANY CHARACTER TO ABORT WHILE
                         RUNNING.
     00015           **
     00016           *   CONTROL RETURNS TO PROM MONITOR.
     00017           **
     00018 00F3          ORG     $F3
     00019 00F3          FCB     $FF TURN OFF TTY ECHO
     00020           *
```

DMP PAGE 00002

```
 ER. LINE. LOC.  VALUE.   INPUT.

     00021              * MONITOR ROUTINES.
     00022              * ADDRESSES ARE FOR ACIA VERSION OF
                          MONITOR.
     00023              *
     00024        FF81  OUTCH  EQU @177601
     00025        FF6D  OUT2H  EQU @177555
     00026        FF62  BADDR  EQU @177542
     00027        FF82  OUTS   EQU @177602
     00028        FFAB  MONIT  EQU @177653
     00029        FF24  POLCAT EQU @177444
     00030 0000         ORG 0
     00031 0000  0001  XHI    RMB 1       TEMP FOR HIGH BYTE OF X
     00032 0001  0001  XLO    RMB 1       TEMP FOR LOW BYTE OF X
     00033 0002  0002  LSTBYT RMB 2       ADDRESS OF LAST BYTE
     00034 0004  0001  COUNT  RMB 1       COLUMN COUNTER
     00035 0005  8D40  GO     BSR GETADR  GET FIRST ADDR
     00036 0007  DF00         STX XHI     STORE IT
     00037 0009  8D3C         BSR GETADR  GET LAST ADDR
     00038 000B  08           INX         ADJUST IT
     00039 000C  DF02         STX LSTBYT  STORE IT
     00040 000E  DE00         LDX XHI     POINT TO FIRST BYTE
```

DMP PAGE 00003

```
 ER. LINE. LOC.  VALUE.    INPUT.

     00041 0010  C60D     CRLF   LDA B  #@15   SEND CRLF
     00042 0012  BDFF 81         JSR    OUTCH
     00043 0015  C60A            LDA B  #@12
     00044 0017  BDFF 81         JSR    OUTCH
     00045 001A  C611            LDA B  #17
     00046 001C  D704            STA B  COUNT  INIT COUNTER
     00047 001E  DF00            STX    XHI    PRINT ADDR
     oo048 0020  9600            LDA A  XHI
     00049 0022  BDFF 6D         JSR    OUT2H
     00050 0025  9601            LDA A  XLO
     00051 0027  BDFF 6D         JSR    OUT2H
     00052 002A  7A00 04 NXTBYT  DEC    COUNT
     00053 002D  27E1            BEQ    CRLF
     00054 002F  BDFF 82         JSR    OUTS   SEND A SPACE
     00055 0032  A600            LDA    X      BYTE TO A
     00056 0034  BDFF 6D         JSR    OUTZH  PRINT IT
     00057 0037  08              INX           BUMP POINTER
     00058 0038  9C02            CPX    LSTBYT ARE WE DONE
     00059 003A  2708            BEQ    JMONIT YES,RETURN TO
                                                MONITOR
     00060 003C  BDFF 24         JSR    POLCAT NO, WANT TO QUIT
```

```
 ER. LINE. LOC.  VALUE.    INPUT.

     00061 003F  24E9           BCC NXTBYT
     00062 0041  B6F0 01        LDA $F001 YES, READ CHAR
     00063 0044  7EFF AB JMONIT JMP MONIT AND RETURN TO MONITOR
     00064               *
     00065               * GETADR LOADS X WITH ADDRESS
     00066               * READ FROM TTY
     00067               *
     00068 0047  BDFF 82 GETADR JSR    OUTS  SEND SPACE
     00069 004A  C63F           LDA B  #'?   SEND QUESTION MARK
     00070 004C  BDFF 81        JSR    OUTCH
     00071 004F  BDFF 62        JSR    BADDR GET ADDRESS
     00072 0052  39             RTS          RETURN
     00073               *
     00074               * RESTORE TTY ECHO AFTER LOAD
     00075               *
     00076 00F3                 ORG    $F3
     00077 00F3                 FCB    00
     00078                      END

TOTAL ERRORS   00000
```

BADDR	FF62
COUNT	0004
CRLF	0010
GETADR	0047
GO	0005
JMONIT	0044
LSTBYT	0002
MONIT	FFAB
NXTBYT	002A
OUT2H	FF6D
OUTCH	FF81
OUTS	FF82
POLCAT	FF24
XHI	0000
XLO	0001

REFERENCES

(1) Sherman, P.M. Programming and Coding Digital Computers, John Wiley and Sons, N.Y. (1963).

(2) Barron, D. W. Assemblers and Loaders, American Elsevier N.Y. (1971).

(3) Altair 680b Programming Manual, MITS (1976).

(4) Motorola Microprocessor Applications Manual, McGraw-Hill, N.Y. (1975).

(5) Yourdon, E. Techniques of Program Structure and Design, Prentice Hall, N.J. (1975).

(6) Kernighan, B.W., and Plauger, P.J. The Elements of Programming Style, McGraw-Hill, N.Y. (1974).

(7) Donovan, J.J. Systems Programming, McGraw-Hill, Tokyo, (1972).

(8) Knuth, D.E. The Art of Computer Programming, Volume 3, Sorting and Searching, Addison-Wesley, Publishing Company, Reading, Mass. (1973).