ALTAIR 680B CROSS-ASSEMBLER
ALTAIR 6808 CROSS-ASSEMBLER

by

PRADHEEP S. KUMAR B. Tech.

A Project
Submitted to the School of Graduate Studies
in Partial Fulfilment of the Requirements
for the Degree
Master of Science
McMaster University
February 1978
MASTER OF SCIENCE (1978) McMaster University
(Computation) 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

<table>
<thead>
<tr>
<th>Chapter I: INTRODUCTION</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Machine Language</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Assembly Language</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Overview of the Cross-Assembler</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Altair 680b Micro-computer</td>
<td>4</td>
</tr>
<tr>
<td>1.5 Approach</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter II: THE ASSEMBLER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Introduction</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Functions</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Data Structures</td>
<td>10</td>
</tr>
<tr>
<td>2.4 General Algorithm</td>
<td>13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter III: FIRST PASS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Introduction</td>
<td>15</td>
</tr>
<tr>
<td>3.2 Machine-instruction processing</td>
<td>15</td>
</tr>
<tr>
<td>3.3 Pseudo-instruction processing</td>
<td>19</td>
</tr>
<tr>
<td>3.4 Conclusion</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter IV: SECOND PASS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Introduction</td>
<td>22</td>
</tr>
<tr>
<td>4.2 Machine-instruction processing</td>
<td>22</td>
</tr>
<tr>
<td>4.3 Pseudo-instruction processing</td>
<td>26</td>
</tr>
<tr>
<td>4.4 Conclusion</td>
<td>27</td>
</tr>
</tbody>
</table>
Chapter V: TABLE-HANDLING

5.1 Introduction 29

5.2 Symbol-table searching 29

5.3 Symbol-table sorting 32

5.4 Machine-operation-table searching 32

CHAPTER VI: CONCLUSION 36

APPENDICES

A - Instructions to User 37

B - Error Messages 38

C - Absolute-Binary Format 45

D - I-O Interface 47

E - Language Description 48

F - Grouped Listing 51

G - Pseudo-Instructions 52

H - Program Listing 55

I - Sample Programs 270

REFERENCES 289
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. 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 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 microcomputer, 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 a 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

address bus

MPU

MEMORY

I/O DEVICES
data bus

7
ACCA
Accumulator A
7
ACCB
Accumulator B
15
IX
Index register
15
PC
Program counter
15
ST
Stack pointer
5
Condition code register

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."

"Write clearly—don't be too clever". 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 databases used are:

(1) Source-code line--INS LN

(2) Pseudo-operation table--POPTB

(3) Machine operation table--MOPTB

(4) Storage required by current instruction--LENT H

(5) Location-counter--LOC TR

(6) Symbol-table--SYMTB

(7) Error-Tables--ER1TB, ER2TB

(8) Direct-address-table--DIRAD

(9) Print-line--PRTLN

(10) Object-code buffer--B NOUT

INS LN 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 INS LN 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 up to 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**

<table>
<thead>
<tr>
<th>4F</th>
<th>50</th>
<th>54</th>
<th>00</th>
<th>02</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>P</td>
<td>T</td>
<td>null</td>
<td>service routine address</td>
<td></td>
</tr>
</tbody>
</table>

### MACHINE-OP TABLE ENTRY

**Fig. 2-2**

<table>
<thead>
<tr>
<th>54</th>
<th>41</th>
<th>42</th>
<th>00</th>
<th>08</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>A</td>
<td>B</td>
<td>null</td>
<td>type</td>
<td>base value</td>
</tr>
</tbody>
</table>

### SYMBOL TABLE ENTRY

**Fig. 2-3**

<table>
<thead>
<tr>
<th>53</th>
<th>50</th>
<th>4D</th>
<th>42</th>
<th>4F</th>
<th>4C</th>
<th>00</th>
<th>E1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 up to 50 entries.

BNOUT and PRTLN are the object code buffer and the print-line buffer respectively. PRTLN can accommodate up to 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
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.
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-
START

Initialise, Punch Header record

Initialise

Read next line

Update counters

1

comment

No

Get op-code

machine-op

opcode

pseudo-op

SFCOND PASS

Fig. 4-1
Fig. 4-1 (contd)
Set LENTH. Assemble instruction.

Set LENTH. Assemble instruction.

Set LENTH. Assemble instruction.

Enter into object-code buffer

Print source listing

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 flag­
ged as an error.
FCC:
This pseudo-op generates 7-bit ASCII character constants. Basic­
ally, there are two ways of defining character strings--by using de­
limiters 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 eval­
uated, 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,

\[
   h(k) = \left\lfloor M((A/w \times k) \mod 1) \right\rfloor
\]

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 \( K \)
START

i ← h₁(k)

Table (i) empty?

Yes → Overflow Stop

No → N = M - 1

Yes → K = Key(i)

No → N ← N + 1

Overflow Stop

Key(i) ← K

Unsuccessful Stop

Successful Stop

SYMBOL TABLE SEARCHING

Fig. 5-1
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
START

Set left-most bit of every null entry to 1.

Get minimum entry

null

YES

Set left-most bit of all entries to 0

STOP

NO

Print entry

Set left-most bit to 1.

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' $11464_8$
$1 \times 000_8$ thro' $15060_8$
APPENDIX - B

ERROR MESSAGES

<table>
<thead>
<tr>
<th>CODE</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>Delimiter not found in FCC</td>
</tr>
<tr>
<td>E</td>
<td>Illegal expression</td>
</tr>
<tr>
<td>F</td>
<td>Format error</td>
</tr>
<tr>
<td>I</td>
<td>Invalid Label</td>
</tr>
<tr>
<td>L</td>
<td>String too long in FCC</td>
</tr>
<tr>
<td>M</td>
<td>Multiply defined label</td>
</tr>
<tr>
<td>N</td>
<td>Label not present in EQU</td>
</tr>
<tr>
<td>O</td>
<td>Opcode unrecognizable</td>
</tr>
<tr>
<td>P</td>
<td>Label present in ORG, END</td>
</tr>
<tr>
<td>R</td>
<td>Expression cannot be evaluated in pass-1, ORG, RMB</td>
</tr>
<tr>
<td>S</td>
<td>Symbol too long</td>
</tr>
<tr>
<td>U</td>
<td>Undefined Symbol</td>
</tr>
<tr>
<td>V</td>
<td>Value overflow, division by 0.</td>
</tr>
</tbody>
</table>
PROGRAMS WITH ERRORS
ERR1  

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
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

ER. LINE. LOC. VALUE. INPUT.

F 00021 0006  JMPA   $10
F 00022 0006  BCCA   $10
F 00023 0006  AAK
F 00024 0006  *
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
<table>
<thead>
<tr>
<th>ER.</th>
<th>LINE</th>
<th>LOC</th>
<th>VALUE</th>
<th>INPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>00001</td>
<td>NAM</td>
<td>ERR2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00002</td>
<td>*</td>
<td>ERROR-CODES TEST.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00003</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00004</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00005</td>
<td>***** INVALID LABEL, CODE-I *****</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00006</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>00007 0000</td>
<td>16</td>
<td>A</td>
<td>TAB</td>
</tr>
<tr>
<td>I</td>
<td>00008 0001</td>
<td>16</td>
<td>B</td>
<td>TAB</td>
</tr>
<tr>
<td>I</td>
<td>00009 0002</td>
<td>16</td>
<td>X</td>
<td>TAB</td>
</tr>
<tr>
<td>I</td>
<td>00010 0003</td>
<td>16</td>
<td>1AC</td>
<td>TAB</td>
</tr>
<tr>
<td>I</td>
<td>00011 0004</td>
<td>16</td>
<td>ABCDEFG</td>
<td>TAB</td>
</tr>
<tr>
<td>I</td>
<td>00012 0005</td>
<td>16</td>
<td>AB$</td>
<td>TAB</td>
</tr>
<tr>
<td>00013</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00014</td>
<td>***** MULTIPLY DEFINED LABEL, CODE-M *****</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00015</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>00016 0006</td>
<td>16</td>
<td>KKK</td>
<td>TAB</td>
</tr>
<tr>
<td>00017 0007</td>
<td>16</td>
<td>KKK</td>
<td>TAB</td>
<td></td>
</tr>
<tr>
<td>00018</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00019</td>
<td>***** DELIMITER NOT FOUND, CODE-D *****</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00020</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ERR2

<table>
<thead>
<tr>
<th>ER.</th>
<th>LINE.</th>
<th>LOC.</th>
<th>VALUE.</th>
<th>INPUT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>00021</td>
<td>0008</td>
<td>FCC /TEXT/</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D 00022</td>
<td>000C</td>
<td>FCC /TEXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00023</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00024</td>
<td></td>
<td>***** STRING TOO LONG,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE-L *****</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00025</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L 00026</td>
<td>004A</td>
<td>FCC 256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00027</td>
<td>004A</td>
<td>FCC 255,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00028</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00029</td>
<td></td>
<td>***** OPCODE UNRECOGNIZABLE,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE-O *****</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00030</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00031</td>
<td>0149 06</td>
<td>TAP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00032</td>
<td>014A</td>
<td>TT ABC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00033</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00034</td>
<td></td>
<td>***** UNDEFINED SYMBOL,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CODE-U *****</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00035</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U 00036</td>
<td>014A 27B4</td>
<td>BEQ CDE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00037</td>
<td>014C 27B2</td>
<td>BEQ CED*2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U 00038</td>
<td>014E 24B0</td>
<td>ECC X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U 00039</td>
<td>0150</td>
<td>PPCCO 00</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U 00040</td>
<td>0153</td>
<td>7E00 00</td>
<td>STX A $10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>JMP $#10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ERR2
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
  00050 015C 27A2 BEQ %112*$3
  00051 *
  00052 ***** VALUE OVERFLOW, CODE-V *****
  00053 *

V 00054 015E 27A0 BEQ $10/0
  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.

S0 = 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.
s = start of record
cc = Type of record

Byte count
Address
Data

MOTOROLA PAPER TAPE FORMAT
Fig. C-1
APPENDIX - D

I-O INTERFACE

The various I-O 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  - Source 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-O changes can be implemented by modifying these 2 routines appropriately.
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 cor­
responding 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

PSUEDO-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
<table>
<thead>
<tr>
<th>SUBROUTINE</th>
<th>PAGE</th>
<th>SUBROUTINE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABA71</td>
<td>156</td>
<td>FNE26</td>
<td>92</td>
</tr>
<tr>
<td>ADC63</td>
<td>204</td>
<td>ENET9</td>
<td>111</td>
</tr>
<tr>
<td>ADD88</td>
<td>201</td>
<td>EPT83</td>
<td>143</td>
</tr>
<tr>
<td>ASL65</td>
<td>215</td>
<td>EQU53</td>
<td>210</td>
</tr>
<tr>
<td>ASSMB</td>
<td>64</td>
<td>FRL28</td>
<td>145</td>
</tr>
<tr>
<td>BCC70</td>
<td>190</td>
<td>ESC32</td>
<td>141</td>
</tr>
<tr>
<td>BDC46</td>
<td>234</td>
<td>FVL97</td>
<td>180</td>
</tr>
<tr>
<td>BHX47</td>
<td>138</td>
<td>EWS31</td>
<td>237</td>
</tr>
<tr>
<td>BIN15</td>
<td>157</td>
<td>RX802</td>
<td>192</td>
</tr>
<tr>
<td>BNY93</td>
<td>170</td>
<td>FCB54</td>
<td>229</td>
</tr>
<tr>
<td>CAC22</td>
<td>85</td>
<td>FCC55</td>
<td>226</td>
</tr>
<tr>
<td>CER14</td>
<td>212</td>
<td>FDB56</td>
<td>231</td>
</tr>
<tr>
<td>CHCL6</td>
<td>83</td>
<td>FVR34</td>
<td>254</td>
</tr>
<tr>
<td>CHK73</td>
<td>82</td>
<td>GCH81</td>
<td>118</td>
</tr>
<tr>
<td>CLB76</td>
<td>87</td>
<td>GET78</td>
<td>96</td>
</tr>
<tr>
<td>CPX67</td>
<td>219</td>
<td>GLB41</td>
<td>114</td>
</tr>
<tr>
<td>CVT84</td>
<td>149</td>
<td>GNC87</td>
<td>168</td>
</tr>
<tr>
<td>CVT96</td>
<td>184</td>
<td>HEX94</td>
<td>171</td>
</tr>
<tr>
<td>DEC91</td>
<td>175</td>
<td>HSH80</td>
<td>102</td>
</tr>
<tr>
<td>DI800</td>
<td>193</td>
<td>IMM99</td>
<td>188</td>
</tr>
<tr>
<td>DIV90</td>
<td>202</td>
<td>IBN35</td>
<td>159</td>
</tr>
<tr>
<td>EBN36</td>
<td>161</td>
<td>IN801</td>
<td>196</td>
</tr>
<tr>
<td>ELC29</td>
<td>137</td>
<td>INIT0</td>
<td>65</td>
</tr>
<tr>
<td>END52</td>
<td>155</td>
<td>INPS4</td>
<td>76</td>
</tr>
<tr>
<td>SUBROUTINE</td>
<td>PAGE</td>
<td>SUBROUTINE</td>
<td>PAGE</td>
</tr>
<tr>
<td>------------</td>
<td>------</td>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td>INT13</td>
<td>70</td>
<td>PSY39</td>
<td>261</td>
</tr>
<tr>
<td>JMP69</td>
<td>223</td>
<td>PUN40</td>
<td>166</td>
</tr>
<tr>
<td>LAC21</td>
<td>86</td>
<td>PUT75</td>
<td>95</td>
</tr>
<tr>
<td>LOC30</td>
<td>148</td>
<td>RED74</td>
<td>94</td>
</tr>
<tr>
<td>LSM18</td>
<td>256</td>
<td>RMB61</td>
<td>206</td>
</tr>
<tr>
<td>MOD85</td>
<td>152</td>
<td>RNCDS5</td>
<td>79</td>
</tr>
<tr>
<td>MPY72</td>
<td>186</td>
<td>SAB98</td>
<td>199</td>
</tr>
<tr>
<td>NAM57</td>
<td>236</td>
<td>SCP23</td>
<td>115</td>
</tr>
<tr>
<td>NER17</td>
<td>249</td>
<td>SMT44</td>
<td>132</td>
</tr>
<tr>
<td>OCT92</td>
<td>173</td>
<td>SPC62</td>
<td>241</td>
</tr>
<tr>
<td>ORG59</td>
<td>208</td>
<td>SPM25</td>
<td>124</td>
</tr>
<tr>
<td>ORR77</td>
<td>109</td>
<td>SPT43</td>
<td>127</td>
</tr>
<tr>
<td>PAG60</td>
<td>245</td>
<td>SST51</td>
<td>97</td>
</tr>
<tr>
<td>PASS1</td>
<td>69</td>
<td>ST804</td>
<td>224</td>
</tr>
<tr>
<td>PASS2</td>
<td>73</td>
<td>STA64</td>
<td>213</td>
</tr>
<tr>
<td>PCH37</td>
<td>164</td>
<td>STB79</td>
<td>103</td>
</tr>
<tr>
<td>PED20</td>
<td>240</td>
<td>STS68</td>
<td>221</td>
</tr>
<tr>
<td>PLF08</td>
<td>251</td>
<td>SUB89</td>
<td>202</td>
</tr>
<tr>
<td>PN803</td>
<td>243</td>
<td>SUM50</td>
<td>163</td>
</tr>
<tr>
<td>PNL12</td>
<td>253</td>
<td>SYM95</td>
<td>177</td>
</tr>
<tr>
<td>PROP7</td>
<td>117</td>
<td>UPD11</td>
<td>110</td>
</tr>
<tr>
<td>PRT82</td>
<td>129</td>
<td>VLB42</td>
<td>89</td>
</tr>
<tr>
<td>PSC10</td>
<td>135</td>
<td>VTR24</td>
<td>120</td>
</tr>
<tr>
<td>PSH66</td>
<td>217</td>
<td>WDUP3</td>
<td>75</td>
</tr>
</tbody>
</table>

WRT86........  151
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFLAG</td>
<td>Flag is 1 for currently filling output buffer, 0 for currently reading input line</td>
</tr>
<tr>
<td>ALVL</td>
<td>Blank is 1 for line containing object code, 0 for comment line</td>
</tr>
<tr>
<td>CNTRL</td>
<td>Blank is 1 for current position is at end of input line, 0 for any other position</td>
</tr>
<tr>
<td>CIP</td>
<td>Blank is 1 for current position is at end of input line, 0 for any other position</td>
</tr>
<tr>
<td>DIFAD</td>
<td>Blank is 1 for directive is an operand of #define, 0 for any other type of directive</td>
</tr>
<tr>
<td>DEPT</td>
<td>Blank is 1 for last entry in directory, 0 for any other position</td>
</tr>
<tr>
<td>EPT</td>
<td>Blank is 1 for last entry of zero test, 0 for any other position</td>
</tr>
</tbody>
</table>

OTHER: 11
SOURCE: 11
SOURCE LISTING: 11
OUTPUT DEVICE: 11
SOURCE CODE: 11
INPUT DEVICE: 1
SOURCE LISTING: 1
OUTPUT DEVICE: 1
ENCFG:  BLK  1  
ENRG:  BLK  1  
ERFB:  ERIT1  
ERSS:  ERIT1  
ENAV:  BLK  1  
INSL1:  
LFLG1:  BLK  1  
LOFLG:  BLK  1  
LTH1:  BLK  1  
LCT1:  BLK  1  
LOCAL:  BLK  1  
LCCTH:  BLK  1  
LCCTR:  BLK  1  
LTFLG:  BLK  1  
MCRT1:  NAME1  
NAME1:  
CORE:  .111  
NOSYM:  BLK  1  
NOMS:  25  
CPB1G:  BLK  1  
CPDF1:  CPDF1  
CPFFT:  CPFFT1  
CPPFG:  BLK  1  

MD.B PASS ID
WO.THER CURR.
PAS. HAS ENDED
ADD. OF PASS1
CORR. TABLE
ADD. OF PASS2
CORR. TABLE
LAP. V.
ADD. OF INPU
LINE
LAP. LABEL
IS PRESENT IN
CURR. LINE
WHETHER LCCTR
HAS TO BE
LISTED
LENGTH OF CURR.
ENT INST.
CURR. LINE NO.
NO. OF VGS. IN
OBJECT OFFSET
VALUE ENCODED
IN THE LOC.
FIELD
ALL. OF CURR.
INSTRUCTION
NAME THIS LISTING
IS GOOD
AUX. OF LOC.
AUX. OF CURR.
NAME
AUX. OF ENTRIES
IN. LIST
NO. OF SYMBOLS
IN SYM. TABLE
NO. OF VGS.
PER DATA SECT.
POINTS TO OPSE
IN. INPUT LINE
ADD. OF DEF.
OPTIONS TABLE
ADD. OF CURR.
OPTIONS TABLE
POINTS TO OPSE
CHANS IN
INPUT LINE.
The following represent the addresses of various routines used, so that they can be accessed indirectly through the base page.
<table>
<thead>
<tr>
<th>Address</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>AC135</td>
</tr>
<tr>
<td>0002</td>
<td>AC135</td>
</tr>
<tr>
<td>0003</td>
<td>AC135</td>
</tr>
<tr>
<td>0004</td>
<td>AC135</td>
</tr>
<tr>
<td>0005</td>
<td>AC135</td>
</tr>
<tr>
<td>0006</td>
<td>AC135</td>
</tr>
<tr>
<td>0007</td>
<td>AC135</td>
</tr>
<tr>
<td>0008</td>
<td>AC135</td>
</tr>
<tr>
<td>0009</td>
<td>AC135</td>
</tr>
<tr>
<td>000A</td>
<td>AC135</td>
</tr>
<tr>
<td>000B</td>
<td>AC135</td>
</tr>
<tr>
<td>000C</td>
<td>AC135</td>
</tr>
<tr>
<td>000D</td>
<td>AC135</td>
</tr>
<tr>
<td>000E</td>
<td>AC135</td>
</tr>
<tr>
<td>000F</td>
<td>AC135</td>
</tr>
</tbody>
</table>
The following represent the addresses of the service routines for the nine different types of machine-operation codes of the M8001.

<table>
<thead>
<tr>
<th>Service Routine</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC63</td>
<td>AC163</td>
</tr>
<tr>
<td>STIA64</td>
<td>ST164</td>
</tr>
<tr>
<td>ASL65</td>
<td>AS165</td>
</tr>
<tr>
<td>PSH66</td>
<td>PS166</td>
</tr>
<tr>
<td>CPX67</td>
<td>CP167</td>
</tr>
<tr>
<td>STS68</td>
<td>ST168</td>
</tr>
<tr>
<td>MP866</td>
<td>JM169</td>
</tr>
<tr>
<td>BCC76</td>
<td>BC170</td>
</tr>
<tr>
<td>AB4711</td>
<td>AB171</td>
</tr>
</tbody>
</table>

.loc 2000

### NAME

**Function:** ASSMB

**Input:** SOURCE-CODE AS DEFINED IN THE REPORT.

**Output:** SOURCE-LISTING, OBJECT CODE IN THE MOTOROLA PAPER-TAPE FORMAT.

**Calls:** MODE5, PASS1, PASS2

**Called-by:** NONE

**Global-variables-used:** SENDAD.

**Global-variables-changed:**
ERROR - BITS-SET.

MODES IS CALLED TO DECIDE PASS1 OR PASS2 IS TO BE CALLED. MODES ALSO SETS FLAGS TO DETECT WHETHER OBJECT CODE OR LISTING IS NECC.

Asm:

ENDAD=1, PASS2 IS NECC
JMP NEXT

ENDAD=1, PASS1
JSR PASS1
JMP ASSM

PASS2.

NEXT:
JSR PASS2
JMP ASSM

NAME.
FUNCTION.
INIT.

INITIALISES VARIABLES BEFORE PASS1.

INPUT.

NONE.

OUTPUT.

NONE.

CALLS.

NONE.

CALLED-BY.

PASS1.

GLOBAL-VARIABLES-USED.
DIRAD, DEF, FER1, FER2, NAME, OPTE, OP11, SRT1.
GLOBAL VARIABLES CHANGED

ERROR-BITS-SET

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
CN300: *BLK 1
CN400: 6
CN500: *BLK 1
CN600: 1
CN700: 5
CN800: 3
CN900: 4
CTR00: *BLK 1

SAVE ACCS., ENTRY

IN100: STA J; AC000
       STA 1; AC100
       STA 2; AC200
       STA 3; AC300

INITIALISE BLFLG, LOCRT, LTFLG TO 0.
INITIALISE ENDFG, LOCSTN, TO 0.

STA 0; AC000
STA 0; BLFLG
STA 0; ENDFG
STA 0; LOCSTN
STA 0; LOCRT
STA 0; LTFLG
INITIALISE CIRAD, WRITE TO ALL ZEROS.

LDA 3, CH200
STA 3, CH800
LDA 2, D1AC
LDA 3, E1TB

STA 0, J, 2
STA 2, J, 3
INC 2, J
INC 3, J
JSZ CH31C
JMP *:

INITIALISE LNOTR, PSFLG, TO 1.

LDA 3, CH100
STA 0, LNOTR
STA 0, PSFLG

INITIALISE EGET.

LDA 0, D1AC
STA 0, EGET

INITIALISE E1BT.

LDA 0, E1TB
STA 0, E1BT

INITIALISE NAME TO DEFINE.

LDA 2, DEFINE
LDA 3, NAME
STA 0, 2
LDA 0, 3
STA 1, 2
LDA 0, 1
STA 2, 2
LDA 0, 2
STA 0, 3

INITIALISE OPTION FLAG TABLE (CPFTB) TO DEFAULT VALUES (CPDTB).
; CN403 CONTAINS DEC 6.
LDA 3,CN403
STA 0,CN503
LDA 2,OPT
LDA 3,OPFTB

; LBA 3,2
STA 0,2
INC 2
INC 3
DSZ CN503
JMP -5

; INITIALISE NOSYM.
; CN503 HAS VALUE DECIMAL 3;
LDA 2,OPTE
LDA 0,CHJU
LDA 0,CHJU
LDA 0,CHJU
STA 0,NOSYM

; INITIALISE SYMBOL TABLE, 256 ENTRIES, FIRST WORD OF EACH ENTRY IS CLEARED.
LDA 0,CYN06
STA 0,CTRL

LDA 3,SYNT
LDA 1,01,906
LDA 0,CYN06

; VALUE 6
STA 3,4,5
ADC 1,4
DSZ CTRL
JMP -5

; RESTORE ACO 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.         INITIAL, INPUT, MACCR, CHCLE,
                PROP7, ENET, PSOLJ, UPDII,
                ERE17.
CALLED-BY.     ASSEMBLY.
GLOBAL-VARIABLES-USED.  ASMX3, ENDPG.
GLOBAL-VARIABLES-CHANGED. NONE.
ERROR-BITS-SET.  NONE.

LOCATIONS FOR STORING ACC CONTENTS.
AC0011  6BLK  1
AC1011  6BLK  1
AC2011  6BLK  1
AC3011  6BLK  1

STORE ACC CONTENTS.
PA1011 STA 1,AC0011
        STA 1,AC1011
        STA 2,AC2011
        STA 3,AC3011

INITIALISE ACC. VARIABLES.
        J8R= INITIAL
CALL RECC ROUTINES UNTIL END PSEUDO-UP IS ENCOUNTERED.

LB101: JSR< INPUT
JSR< ENCOD
JSR< CHCOL

CHECK WHETHER COMMENT OR NOT

LDA J,CMFLG
MOV 0,J,SNR

NOT A COMMENT.

JSR< PROP7
JSR< ENET9
JSR< PSC10
JSR< UPD11

CHECK FOR END OF PASS.

LDA 0,ENDFG
MOV 0,J,SNR
JMP LE101

END OF PASS-1.
LIST OUT TOTAL NO. OF ERRORS.

JSR< NER17

RESTORE ACCS. AND RETURN.

LDA 0,AC101
LDA 1,AC101
LDA 2,AC201
JMP< AC101

NAME:       INITI3
FUNCTION: INITIALISES THE RECC. VARIA
INPUT.
OUTPUT.
CALLS.
COLLECT-BY.
GLOBAL-VARIABLES-USED.
GLOBAL-VARIABLES-CHANGED.
ERROR-BITS-SET.
SPACE FOR SAVING ACOS.
ACC13:  3LK  1
ACC15:  3LK  1
ACC19:  3LK  1
ACC33:  3LK  1

CONSTANTS.
CN13:   52  DEC. 40
CN33:   62  DEC. 52
CTT3:   3LK  1

SAVE ACOS. ENTRY.
IN13:   STA  J,ACI3
        STA  1,ACL5
        STA  2,AC23
        STA  3,AC35

INITIALISE E2ST.
        LJG  W,ER2TB
        STA  0,E2ST

INITIALISE LCTR,LOCAL,LOCN,FAGNO,ENCFG TO 0.
        SUB  0,0
STA 0, Lokal
STA 0, LOCT
STA 0, PAGNO
STA 0, ENGFG

INITIALISE LNCTR TO 1.

INC 0, J
STA 0, LNCT

INITIALISE FSFLG TO 2.

INC J, J
STA 0, FSFLG

INITIALISE EF2TE, CH113=4 J.

LOA J, CH113
STA J, CT<13

LOA 2, EF2TE
SUE 0, 0

LB113: STA J, J, 2
INC JSZ, CT<13
JMP LB113

INITIALISE ENOUT, CH113=4 J.

LOA J, CH113
STA J, CTR<13

LOA 2, ENOUT
SUE 0, 0

LB213: STA J, J, 2
INC JSZ, CT<13
JMP LB213

CALL ON ROUTINE TO PUNCH OUT START RECORD.

JSPS PN104
CALL ON ROUTINE TO PRINT OUT HEADING.
USES PAGE6

RESTORE ACCS AND RETURN.
LDA 0,AC013
LDA 1,AC113
LDA 2,AC213
JMP  AC313

NAME.
FUNCTION.
PRODUCTS SOURCE-LISTING,
OBJECT CODE,SYMBOL TABLE.

INPUT.
SYMBOL TABLE, SOURCE-CODE,
#DIRECT-ADDRESS TABLE.

OUTPUT.
OBJECT CODE, SOURCE-LISTING,
EXPER LISTING, SYMBOL TABLE.

CALLS.
LAT, INPS4, MNC5, CCHL6,
PKCP7, DEP14, ENET3, B1M15,
PSC10, UP011, RDU03.

CALLED-BY.
ASSMB.

GLOBAL-VARIABLES-USED.
SMFLG, ENDFG.

GLOBAL-VARIABLES-CHANGED.
NONE.

ERROR-BITS-SET.
NONE.

LOCATIONS FOR STORING ACC CONTENTS.

AC0021, BLK 1
AC1021, BLK 1
AC2021, BLK 1
AC3021, BLK 1
SAVE ACC CONTENTS.

PA162:
STA 0, AC102
STA 1, AC102
STA 2, AC302
STA 3, AC302

INITIALISE REG. VARIABLES.
JSR # INT13

SCAN SOURCE CODE LINE BY LINE UNTIL END PSEUDOC-OP IS ENCOUNTERED.

LEL02:
JSR # INPS4

READ FROM INPUT
JSR # RK005

CHECK SOL 1 AND TAKE ACTION.
JSR # CH016

CHECK WHETHER COMMENT OR NOT.

LDA #6, CHFLG
MOV U, J, SR

NOT A COMMENT, PROCESS OH-CODE, CHECK EACH WORD, ENTER INTO ERROR TABLE.

JSR # PROP7
JSR # CER14
JSR # ER15

CALL ON ROUTINES TO PRODUCE OBJECT CODE/LISTING.
JSR # BIN15
JSR # PSC11

UPDATE COUNTERS.
JSR # UPD11
CHECK FOR END OF PASS
LDA 0,ENDOFG
JGV 0,1,SHF
JMP LELO2

END OF PASS 2,CLEAN-UP,RESTORE ACO$ AND RETURN.
JSPS WCUPS
LDA 0,AC102
LDA 1,AC102
LDA 2,AC202
JMPS AC302

NAME.
FUNCTION.
INPUT.
OUTPUT.
CALLS.
CALLED-BY.
GLOBAL-VARIABLES-USED.
GLOBAL-VARIABLES-CHANGED.
ERROR-BITS-SET.

DOES THE CLEANING UP.
SYMBOL TABLE,ERROR TABLE.
EOF RECORD,S SYMBOL TABLE,
NO. OF ERRORS.

LOCATIONS TO STORE ACC CONTENTS.
AC303: .BLK 1
SAVE ACC CONTENTS
WD103: STA 3,AC303
PUNCH OUT ECF RECORD.
JSRS PE0320

PRINT OUT TOTAL NO. OF ERRORS.
JSRS M1617

LIST SYMBOL TABLE.
JSRS LE010

RESTORE AC'S AND RETURN.
JSRS AC0353

<table>
<thead>
<tr>
<th>NAME</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>INITIALIZE VARIABLES ON</td>
</tr>
<tr>
<td></td>
<td>EACH SCAN.</td>
</tr>
<tr>
<td>INPUT.</td>
<td>NONE.</td>
</tr>
<tr>
<td>OUTPUT.</td>
<td>NONE.</td>
</tr>
<tr>
<td>CALLS.</td>
<td>NONE.</td>
</tr>
<tr>
<td>CALLED-BY.</td>
<td>PASS1, PASS2.</td>
</tr>
<tr>
<td>GLOBAL-VARIABLES-CHANGED.</td>
<td>ABFLG, BFLG, CFTR, CnFEG, CFNLG, INL1, LAFNLG, LFCFLG, LENDH, UPHED, UPPED, P1E1P, PWFNLG, PFRP, PFRTR, PFTL3, SCFLG, VALUE, VFLG, WSPC1</td>
</tr>
<tr>
<td>GLOBAL-VARIABLES-USED.</td>
<td>WSPACE, PFTL1, INHL1</td>
</tr>
<tr>
<td>ERROR-BITS-SET.</td>
<td>NONE.</td>
</tr>
</tbody>
</table>

SPACE FOR SAVING AC'S CONTENTS.
AC1641,  BLK  1
AC1641,  BLK  1
AC2041,  BLK  1
AC3041,  BLK  1

CONSTANTS, COUNTERS:
CN2441,  2
CN3241,  56
CN4041,  74
CN6441,  2
CT3241,  3

SAVE ACC CONTENTS:
IN1641, STA 0, AC04
STA 1, AC14
STA 2, AC24
STA 3, AC34

INITIALISE MPFLG, LBFLG, LENTH, PNFLG, PRERR, ABFLG, BSVAL, VLFLG, VALUE, VLFLG TO 0.
SUB 0,  J
STA 0, MPFLG
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 CHPT, CNBEG, CPCTR, CPBEG, CPUG, LCFLS, SCPFLG TO 1.
INC 0,  J
STA 0, CHPT
STA 0, CNBEG
STA 0, CPCTR
STA 0, CPBEG
STA 0, CPUG
STA 0, LCFLS
STA 0, SCPFLG
; INITIALISE NSWCE TO 0.
  LDA 2, NSWCE
  STA 0, 2

; INITIALISE PRTLH TO ALL BLANKS. CH4=40, CH5=40=BLANKS.
  LDA 1, CH4
  STA 1, CH5
  LDA 2, PRTLH
  LDA 2, CH5
  LBA 104
  STA 0, 2
  INC 2, 2
  DSZ CH5
  JMP L8104

; INITIALISE I. SLN TO ALL BLANKS. CH4=40, CH5=40=BLANKS.
  LDA 3, CH4
  STA 0, CH5
  LDA 3, CH5
  LDA 2, I. SLN
  LBA 104
  STA 0, 2
  INC 2, 2
  DSZ CH5
  JMP L8204

; SET VALUE FOR PIERR IF PASS2.CH2=2.
  LDA 3, PSFLG
  LDA 1, CH2
  INC 3, 3
  DSZ CH5
  JMP L8204

; CHECK FOR PASS=2
  SUB 0, 1
  DSZ
  JMP END

; PASS IS 2. SET VALUE FOR PIERR.
  LDA 1, LINCTR
LDA $2,EF:1T
LDA $3,E1EPT

LJMP 3
LJMP 4

JMP FND04

WHOLE TABLE HAS NOT BEEN SEARCHED.

LDA $0,3,0
SUBE $0,1,SNF
JMP FND04
INC 2
INC 2
JMP LB304

FOUND IN TABLE. INITIALISE $1EFF?

ENDO4 INC 2,2
LDA $0,2,2
STA $0,PIERR
JMP ENDO4

NOT FOUND, SET TO 0.

END04 SUB 0,3
STA $0,PIERR

RESET ACC CONTENTS AND RETURN.

END04: LDA $0,AC004
LDA 1,AC104
LDA 2,AC204
JMP $AC304

-----------

NAME: ENCES
FUNCTION: GET NEXT CARD FROM INPUT DEVICE INTO INSLEN.
INPUT: NONE.
OUTPUT: None.
CALLS.
CALL-3Y.
GLOBAL-VARIABLES-USED.
GLOBAL-VARIABLES-CHANGED.
ERROR-BITS-SET.

LOCATIONS FOR SAVING ACC CONTENTS.

AC005:  BLK  1
AC105:  BLK  1
AC205:  BLK  1
AC305:  BLK  1

CONSTANTS.  40, BLANK

CH05:    51
BLK05:   40
CTR05:   BLK  1

SAVE ACC CONTENTS.

P105:   STA  0, AC005
         STA  1, AC105
         STA  2, AC205
         STA  3, AC305

SET INPUT AND OUTPUT DEVICE CODES.

LDA  0, SCINC
STA  0, INPOV
LDA  0, EC00V
STA  0, OUT0V

SET COUNTER VALUE.

CM005 = 40 DEC

LDA  0, CM005
STA  0, STR05
LDA  2, INSLN
LE105:  SUB  61J
        JSR  R074
: CLEAR ACC 8.

ACC J HAS CR IN RIGHT HALF.
        JSK  CR73
        JMP  CR105
        MOV  0,1

        JSK  R074
        JSK  CR73
        JMP  CR205
        ADD  1,1
        STA  1,9,2
        INC  1,9,2
        JMP  LE105

COUNT OF 6J CHARACTERS.
        JMP  EN305

ODD CHARACTER IS CR; SC NO CH. TO BE SAVED.
CR105:  JMP  EN305

EVEN CHARACTER IS CR; SAVE 1 CH AND RETURN.
CR205:  MOV  1,1
        LOA  1,9,205
        ADD  1,9
        STA  1,9,2

RESTORE ACC AND RETURN.
END05:  LOA  0,AC005
        LOA  1,AC105
        LOA  2,AC205
        MPS  AC305

NAME.
FUNCTION.

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
ACC 3 # RETURN.
OTHERWISE TAKE
ACC 3 + 1 # RETURN.

CALLS.

PUT75

CALLED BY.

NONE.

GLOBAL-VARIABLES-USED.

NONE.

GLOBAL-VARIABLES-CHANGED.

NONE.

ERROR-BITS-SET.

NONE.

LOCATIONS FOR SAVING ACC CONTENTS.

ACC73: 3LK 1
ACC173: 3LK 1
ACC273: 3LK 1
ACC373: 3LK 1

CONSTANTS.

CH73: 015: CR
CN273: 014: FF
CN373: 012: LF

SAVE ACC CONTENTS.

CH73: STA 0, ACC73
STA 1, ACC173
STA 2, ACC273
STA 3, ACC373

CHECK TO SEE WHETHER CH. IS FF, CR OR LF.
LDA 1, CH173
JUBE 1, SNE
JMP CRT73
; CH. IS CH.

LDA 1, CH273
JUBE 0, SNE
JMP CRT73
; CH. IS FF.

LOA 1, CH373
JUBE 0, SNE
JMP CRT73
; CH. IS LF.

; CH IS NEITHER, TAKE NORMAL RETURN
INC 3, 3
STA 3, AC373
JMP ENO73

; CH IS FF OR CR, PUT OUT A LF AND TAKE AS3 RETURN.
CFT73: LDA 0, CH173
JSRS PCT75

ENO73: LDA 0, AC073
LDA 1, AC173
LOA 2, AC273
JMPS AC373

NAME: CHOLE
FUNCTION: CHECKS COL 1 AND DECIDES IF LINE IS A COMMENT OR LABEL AND TAKES CORR. ACTION
INPUT: INSLN
OUTPUT: NONE
CALLS: LAC21, LAC22, GC31
CALLED-BY: PASS1, PASS2
GLOBAL-VARIABLES-USED: NONE
GLOBAL-VARIABLES-CHANGED: NONE
E- FOR-BITS-SET.

LOCATIONS FOR SAVING ACC CONTENTS.
AC00:  1, BLK  1
AC10:  1, BLK  1
AC20:  1, BLK  1
AC30:  1, BLK  1

CH CODES FOR BLANK, *, AND MASK
CH00:  40  BLANK
CH01:  52  *
CH02:  377  MASK

SAVE ACC CONTENTS.
CH00:  STA  1,AC00
       STA  1,AC10
       STA  1,AC20
       STA  1,AC30

GET FIRST CHARACTER.
JSR$  GC031

ACC CONTAINS FIRST CHARACTER, CHECK THE CHARACTER.
LDA  1,CH00  BLANK
SUBE  0,1,SNR

CHARACTER IS A BLANK, TAKE NO ACTION.
JMP  END00

CHECK WHETHER * OR NOT.
LDA  1,CH00
SUBE  0,1,SNR
JMP  CM005
JMP  CM005

LABEL.
COMMENT ACTION
CMC61 JSPS CAC22

RESTORE ACC AND RETURN.
END61 LDA J, AC036
LDA 1, AC106
LDA 2,AC206
JMPS AC036

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

LOCATIONS FOR STORING ACC CONTENTS.
AC022: .BLK 1
CMC22: 1
CAC22: 1

SAVE ACCS AND SET FLAG.
CA122: STA 0,AC022
       LDA 0,CH022
       STA 0,CMFLG
SET LCFLG TO 0.

SUB 0,JO
STA 0,LCFLG

RESTORE ACC CONTENTS.

LOA 0,AC22
JMP 0,3

NAME.
FUNCTION. LAC21.
SET LCFLG, CHECK LABEL, AND TAKE ACTION.

INPUT.

OUTPUT.

CALLS.

CALLED BY.

GLOBAL-VARIABLES-USED.

GLOBAL-VARIABLES-CHANGED.

ERROR-BITS-SET.

NONE.

NONE.

NONE.

NONE.

LOCATIONS FOR SAVING ACC CONTENTS.

AC21: BLK 1
AC21: BLK 1
AC22: BLK 1
AC22: BLK 1
AC22: BLK 1

CONSTANTS.

NONE.

SAVE ACC CONTENTS.
LA121: STA 0,ACJ21
   STA 1,ACJ21
   STA 2,ACJ22
   STA 3,ACJ21

SET LABEL FLAG.
   LDA 0,CNJ21
   STA 0,LSFL6

GET LABEL. IE. SET CHPR TO POINT TO END
OF LABEL (CH AFTER LABEL).
   JSRS GL341
   CHPR HAS BEEN SET. CALL ON ROUTINE TO CHECK
VALIDITY OF LABEL. AND IF VALID TAKE ACTION.
   JSRS VL342
   FSTORE ACC CONTENTS AND RETURN.
   LDA 0,ACJ21
   LDA 1,ACJ21
   LDA 2,ACJ22
   JMPS ACJ21

NAME: CLB36.
FUNCTION: CHECKS WHETHER CH IN
ACC IS ALPHANUMERIC.
INPUT: CH. IN RIGHT HALF OF ACC C.
OUTPUT: ACC = 1 IF RETURN; IF ALPH.
ACC = 0. RETURN OTHERWISE.
CALLS: NONE.
CALLED-BY: VL342.
GLOBAL-VARIABLES-USED: NONE.
GLOBAL-VARIABLES-CHANGE:
  NONE.
ERROR-BITS-SET:
  NONE.

LOCATIONS FOR STORING ACC CONTENTS:
AC176:
  *BLK  1
AC176:
  *BLK  1
AC376:
  *BLK  1

CONSTANTS:
CH176:
  60
CH176:
  71
CH376:
  101
CH376:
  132

SAVE ACC CONTENTS:
CL176:
  STA  J,AC176
  STA  1,AC176
  STA  2,AC276
  STA  3,AC376

CHARACTER IS IN RIGHT HALF OF ACC:
LOA  1,CH176
SUBLE  1,CH176
    Nxt176
; NOT NUMERIC

LOA  1,CH176
SUBLE  0,CH176
    Nxt176
; NOT NUMERIC
    Vlc176
; VALID CH, NUMERIC

CHECK WHETHER ALPHABETIC:
Nxt176:
LOA  1,CH176
SUBLE  1,CH276
    Er176
; NOT ALPHABETIC, NUMERIC, ERROR

LOA  1,CH376
; ZZ
SUBLE U1, SZC
JMP EAR76
JMP VLD76

VALID CHARACTER, TAKE AC3+1 RETURN.
VLD76: LOA U0, AC376
INC 0, J
STA 0, AC376

EGR76: LOA U0, AC076
LOA 1, AC176
JMP AC376

NAME.
FUNCTION.
INPUT.
OUTPUT.
CALLS.
CALLED-BY.
GLOBAL-VARIABLES-USED.
GLOBAL-VARIABLES-CHANGED.
ERROR-BITS-SET.
LOCATIONS FOR SAVING ACC CONTENTS.

VLD42.
CHECKS VALIDITY OF LABEL
AND TAKES CORR. ACTION.
CURR. POINTS TO C1. AFTER
LABEL.
NONE.
SST51, EMB26, CLB76.
LAC21.
INSLH.
NONE.
BIT 0 (I).
AC1421, SLK 1
AC1421, SLK 1
AC2421, SLK 1

89
AC342:  .BLK 1

: CONSTANTS AND COUNTERS.

CH'42:  10
CH'242: 377
CH'42:  161
CH'42:  132
CT'42:  .BLK 1

: SAVE ALL CONTENTS.

VL142:  STA 0,ACJ42
STA 1,AC142
STA 2,AC242
STA 3,AC342

: CHECK FOR VALIDITY OF LENGTH.

LDA 0,CHPTR
LDA 1,CH'42
SUBLE 1,1,SN2
JMP 1,ER42

; INVALID LENGTH

: CHECK FOR FIRST CHARACTER ALPHABETIC.

LDA 2,INSLH
LDA 0,J42
MVS 0,J42
LCA 1,CHN242
AND 1,0

LDA 1,CHN42
SUBLE 1,1,SN2
JMP 1,ER42
LDA 1,CH42
SUBLE 1,1,SN2
JMP 1,ER42

; NOT ALPHABETIC

: FIRST CH. IS OKAY, CHECK OTHERS FOR ALPHANUMERIC.

STA 0,CHPTR
STA 1,CTR42
SUB 1,CTR42

; SET UP CTR.

LDA 2,INSLH
L61421  LDA  u, j, 2
       LDA  UC, 2
       LDA  u,Ch242
       AND  1, i
               ; MASK

       JSRS  CL376
               ; CHECK FOR
       JMP  ERR42
               ; NOT ALPHA.

       JSZ  CTR42
       JMP  *+2

       LDA  0, g, 2
       LDA  1,Ch242
       AND  1, i
               ; MASK

       JSRS  CL376
       JMP  ERR42
       JSZ  2, 2
       JMP  L6142
               ; ALL ON CHECKED

       LABEL  CHECKED  AND  VALID.

NXT421  JSRS  SS51
       JMP  EN242
       ; ERROR IN LABEL, ENTER IT.

       SUB  0, j
       STA  0, 4FG42
       JSRS  EN25

A66421  , 3LK  1
       ; RESTORE  ACC  AND  RETURN.

       LDA  u, ACJ42
       LDA  1, AC142
       LDA  2, AC142
       JMP  AC542
       ; NAME.
FUNCTION.
INPUT.
OUTPUT.
CALLS.
CALLED-BY.

GLOBAL-VARIABLES-USED.
GLOBAL-VARIABLES-CHANGED.
ERROR-BITS-SET.

LOCATIONS FOR SAVING ACC CONTENTS.
AC026: 6LK 1
AC126: 6LK 1
AC226: 6LK 1
AC326: 6LK 1

CONSTANTS FOR SETTING BIT IN FREQ.
EPS26:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
SAVE ACC CONTENTS.

LOAD FLAG NUMBER.

LOAD CORE VALUE.

#OR# THE BIT IN.

SAVE IT BACK IN FERR.

RESTORE ACC AND RETURN.

---

LOAD 0, J, S
LOAD 2, ERA26
LOAD 0, J, 2

STA 0, AG126
STA 0, J, PERR
STA 0, J, AG226
BLK 1
BLK 1
BLK 1
BLK 1

LOAD J, AG326
STA 0, J, PERR

LOAD J, AG126
LOAD 1, AG126
LOAD 2, AG226
LOAD 3, AG326

JMP 1, 3
NAME.
FUNCTION.
READ NEXT CH AND
ECHO IT OUT.
INPUT.
NONE.
OUTPUT.
IN AC 174 RIGHT
ADJUSTED WITH BIT 8 SET
TO 1.
CALLS.
GET 73, PUT 75.
CALLED BY.
RACES, MODES.
GLOBAL-VARIABLES-USED.
NONE.
GLOBAL-VARIABLES-CHANGED.
NONE.
ERROR-BITS-SET.
NONE.

LOCATIONS FOR SAVING ACC CONTENTS.
AC174: 3BLK 1
AC274: 3BLK 1
AC374: 3BLK 1

SAVE ACC CONTENTS.
FE174: STA 1,AC174
STA 2,AC274
STA 3,AC374

GET NEXT CHARACTER AND PRINT IT OUT
JSPE GET73
JSPE PUT75

RESTORE ACC CONTENTS AND RETURN.
LA 1,AC174
LA 2,AC274
LA 3,AC374
NAME: PUTF5
FUNCTION: Puts out next ch. onto output device (OUTOW).
INPUT: None.
OUTPUT: In ACC & right adjusted.
CALLS: None.
CALLED-BY: LOTS OF ROUTINES.
GLOBAL-VARIABLES-USED: OUTOW.
GLOBAL-VARIABLES-CHANGED: None.
ERROR-BITS-SET: None.

LOCATIONS FOR SAVING ACC CONTENTS:

ACC751: 3BLK 1
C01751: JBLK 1
D002751: 5111JJ
SEN751: 635000

SAVE ACC CONTENTS:
PU1751 STA 1; ACC751
          STA 1; ACC1751

SET UP INST. DEP. ON DEVICE:
LD8 0; OUTOW
10V 1; I515
JMP END751
LR8 1; 00A751
ACC 1; 1
STA 1; I5075

;  LOA  1,50.75
  ADD  ;11
  STA  1,INT 75
;
  LOA  ;,AC 075
  BLK  1
INT  75;  JMP  .-1

RESTORE ACC CONTENTS AND RETURN.

END  75:  LOA  0,AC 075
         LOA  1,AC 175
         JMP  ;,3

NAME:  SET  75
FUNCTION:  GET NEXT CH. FROM
           INPUT DEVICE(INDV)
INPUT:    NONE
OUTPUT:   CH. IN ACC & RIGHT ADJUSTED
          WITH BIT 9 SET TO 1.
CALLS:    NONE
CALLED-BY:  NONE
GLOBAL-VARIABLES-USED:  RECV 14
GLOBAL-VARIABLES-CHANGED:  INPUT.
ERROR-BITS-SET:  NONE

SPACE FOR SAVING ACC CONTENTS.
AC  175:  ;,BLK  1
SAVE ACC CONTENTS
GE178: STA 1,AC178
       LDA 0,INPOV
       JMP EX78
       LDA 1,NI073
       STA 1,IN073
       LDA 1,SO178
       STA 1,IO178
       LDA 1,ST78
       STA 1,STF78

I1C78: *BLK 1
       BLK 1
       JMP -1
I1T78: *BLK 1,SK78
       LDA 1,SK78
       JMP -1,LE78
I1F78: *BLK 1,LC78
       LDA 1,LC78
       JMP IN078
       LDA 0,SNR
       SUBE 0,SNR
       JMP IN078

RESTORE ACC CONTENTS AND RETURN
END78: LDA 1,AC178
       JMP 0,3

NAME.
FUNCTION.
SET:1.
SEARCHED SYMBOL TABLE AND
INSERTS IF ALREADY PRESENT
EXPER BIT IS SET.

INPUT.

OUTPUT.

CALLS.

CALLED-By.

GLOBAL-VARIABLES-USED.

GLOBAL-VARIABLES-CHANGED.

ERROR-BITS-SET.

SPACE FOR SAVING ACC CONTENTS.

ACC511  3BLK  1
ACC521  3BLK  1
ACC531  3BLK  1

CONSTANTS.

CN511  0
CN521  1
CN531  2

M5K511  774UJ
M5K521  400UJ
M5K531  411UJ
M5K541  540UJ

SPACE FOR SAVING LABEL

LBL511  3BLK  3
LBL512  LBL51

SAVE ACC CONTENTS.
SE151:  STA 0, ACJ51
        STA 1, ACJ51
        STA 2, ACJ51
        STA 3, ACJ51

CLEAR LBL51

        LDA 2, LBAS1
        SUB 3, j
        STA j, 0, 2
        STA 0, 0, 2

SET UP INDEX REGISTERS AND COUNTER.

        LDA 2, INSLN
        LDA 3, LBA51
        LDA 0, CHPT6
        STA 0, CT51
        JSZ CT51

ACC 2 CONTAINS STARTING ADDRESS OF LABEL IN INSLN.
ACC 3 CONTAINS STARTING ADDRESS OF LOCAL SPACE FOR LABEL.
CT51 CONTAINS LENGTH OF LABEL.

MOVE LABEL INTO LOCAL SPACE.

HYT51:  LDA 0, 2, 2
        JSZ CT51
        JMP EV151

        LDA 1, MKS51
        AND 1, 0
        STA 0, 2, 3
        JMP OKY51

        STA 0, 3, 3
        INC 3, 3
        INC 3, 3
        JSZ CT51
        JMP HYT51
        JMP OKY51
WHOLE LABEL HAS BEEN TRANSFERRED, LABEL CONTAINING LABEL
FADING ON RIGHT BY HULLS.
CHECK FOR VALIDITY OF LABEL,
( A, X A, OR X )

LDA 0,LBL51
LDA 1,CH51
SUBE 0,SNR
JMP ERR51

LDA 1,CH351
SUBE 0,SNR
JMP ERR51

LDA 1,CHX51
SUBE 0,SNR
JMP ERR51

LABEL IS OKAY.

CALL CH ROUTINE TO CHECK WHETHER LABEL IS PRESENT
OF NOT, AND IF NOT PRESENT, INSERT LABEL.

LDA 0,LBA51
STA 0,AG151

LDA 0,PSFLG
LDA 1,CH151
SUBE 1,
STA 0,AG251

JMP ST379

AG151: BAK 1
AG251: BAK 1
AG351: BAK 1
FLG51: BAK 1

CHECK RETURNED VALUE TO SEE WHETHER FOUND, IF AG351
IS NEGATIVE, NOT FOUND.

LDA 0,FLG51
JGVE 0,SGZC
JMP NDF51

; NOT FOUND.
FOUR.

FNC51:  LDA  0,PSFLG
        LDA  1,CH151
        SUBE  U,CH152
        JMP  EN351 ; CONSTANT 1.
        ; PASS-1.  DO NOTHING.

PASS-1, ENTER INTO BIT 1 OF PDCR.

LDA  0,EN451
JSR<  EN26
AG451:  BLK  1
        JMP  EN351
        ; NOT FOUND IN TABLE.

NFO51:  LDA  0,PSFLG
        LDA  1,CH151
        SUBE  U,SN7
        JMP  EN351 ; CONSTANT 1.
        ; TEST FOR PASS.
        ; PASS-1.  DO NOTHING.

PASS IS 2.

JMP  +1
JMP  +1
JMP  +1
JMP  +1
JMP  +1
JMP  EN351

SPCR R1 LABEL, ENTER AS BIT 0.

EFD51:  SUB  0,0
        STA  0,AG551
        JSR<  EN26
AG651:  BLK  1
        ; RESTORE ACC CONTENTS AND RETURN.

END51:  LDA  0,ACU51
        LDA  1,AC151
        LDA  2,AC251
        JMP<  AC351

101
NAME:  
FUNCTION:  

INPUT:  
OUTPUT:  
ACC 0.
ACC 1.

CALLS:  
NONE.

CALLED-BY:  
STB79.

GLOBAL-VARIABLES-USED:  
OPFTB.

GLOBAL-VARIABLES-CHANGED:  
NONE.

ERROR-BITS-SET:  
NONE.

SPACE FOR SAVING ACC CONTENTS.

AC366:  3BLK 1
AC266:  3BLK 1

SAVE ACC CONTENTS.

HC166:  STA 3,AC366
STA 2,AC266
LDA 3,OPFTB
LDA 2,53

; LOAD SYMBOL TABLE LENGTH.

; CLEAR ACC 1.
; CLEAR CARRY.

LP1-L:  JMP 2,3,SZC
JMP NX760
NAME: TSTNY.

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 1, 
FLAG = 1 DENOTES INSERTION.

OUTPUT: VALUE IN LOCATION ACC 3 + 2, 
NEGATIVE AC IN LOCATION ACC 3 + 3 IF NOT EQU.

CALLS: HSH40, DFR77, HRT36.

CALLED BY: SYM96, SST61.

GLOBAL-VARIABLES-USED: SYMT, UPTF, SLOTO

GLOBAL-VARIABLES-CHANGED: SYMT1, ACLUL, OSYM, BTOV.

ERCF-SETS-SET: none.

CONSTANTS: none.
CH.170:  177777
CH.171:  1
CH.172:  1
KEY70:  1
VIL70:  1
HE70:  1
CH.70:  3

: SPACE FOR SAVING ACC CONTENTS.

ACC70:  1
AC170:  1
AC270:  1
AC370:  1
: SAVE ACC CONTENTS.

ST170:  STA 0,AC170
        STA 1,AC170
        STA 2,AC170
        STA 3,AC170

: CLEAR FLAG.

JUB 0,0
JST 0,1,5

: GET A SINGLE WORD BY CHANGING.

LDA 2,9,3
LDA 0,9,2
LDA 1,1,2

STA 0,AC170
JBRK ORR77
AC170:  1
AC270:  1
AC370:  1

: OR THEM.

LDA 0,AG170
JST 1,AG270
AG170:  1
AG270:  1
AG370:  1

: LOAD RESULT.

LDA 0,AG170
JST 1,AG270
AG170:  1
AG270:  1
AG370:  1

AC470:  1

104
GET MAX VALUE CALCULATED THROUGH
H(k) = (M(A, J1, M+1) * K) MOD 1
WITH A=5, n=LENGTH OF SYMTB, k AS IN A0675, n=10.

LDA 0, A0675  ; K
10Y 5, 1
10C 1, 1
; MPY BY 3.
ACC 0 HAS AK MOD 4.

THE NEXT SET OF INSTRUCTIONS EXTRACT THE MEOC BITS FROM ACC 0, AND PLIES THEN IN ACC 1.

LDS 5 HSH48

ACC 0 CONTAINS THE CHANGED KEY.
ACC 1 CONTAINS H(k).
MULTIPLY BY 4 SINCE EACH ENTRY HAS 4 NOS.

10VL2 1, 1
10VL2 1, 1

STA 0, VAL79
STA 1, KEY79

LDA 3, SYMTB
10C 1, 1
10V 3, 2

LDA 6, 4, 2
10J 6, JNP IF
JMP 2FV79
; LOAD FIRST HD.
; CHK FOR ELY.

CHECK WHETHER KEY MACTHERS.

LJAS 3, AC579
LDA 6, 3, 2
LDA 1, 3, 2
SU3:  J,1,CZP
JMP HCP79

: CHECK

LDA  J,1,3
LDA  J,1,2
JMP HCP79

: NO MATCH

LDA  J,2,3
JMP HCP79

: CHECK

ENTRY FOUND:
JMP EN679

: NO MATCH, REHASH.

HC79:  LDA  0,VALYE
JSR  HSH46

: ECC 1 CONTAINS H(K).
: INVERSE 1 IN BIT 0.

LDA  0,CH376
STA  0,CH776
JSR  OR377

AG79:  3LX  1
AG79:  3LX  1
AG79:  3LX  1

LDA  1,3G376
ICZL  1,1
ICZL  1,1
STA  1,4S279

: HC273 CONTAINS VALUE H2(K).

MT79:  LDA  $EFTE; LOAD EFTE ADDRESS.
LDA  $3,3
LDA  2,3
ICZL  3,3
ICZL  3,3
LDA  1,HC079
LDA  0,KEY79
SUB  1,0
: I=I-H2(K)
LJLE 0,1,SZC
STA 2,1
STA 0,KEY79
SVC 0,3

CHECK IF EMPTY.
LDA 2,SYMB
LDA 3,3
LDA 0,1,2
JMP EPY79

LOAD FIRST NO.

CHECK WHETHER KEY MATCHES.
LDA 3,AC379
LDA 0,1,3
SUBE 0,1,SZC
JMP NMT79
LDA 0,1,3
LDA 1,1,2
SUBE 0,1,SZC
JMP NMT79
LDA 0,2,3
LDA 1,1,3
SUBE 0,1,SZC
JMP NMT79

MATCH FOUND, LOAD LOCK VALUE.
JMP FN379

EMPTY ENTRY, INSERT IF FLAG PERMITS.
ADDRES IS IN AC 2.

EPY79: LDA 0,CHG79
STA 2,ADBL
LDA 3,AC379
STA 0,3,3

SET FLAG.

CHECK FLAG TO SEE IF INSERTION IS REQD.
LDA 0,1,3
LDA 1,0,379
LDA 1,1,379
SUB 0,1,379
JMP EN379

INSERT LABEL.

ICSYM IS A COUNTER FOR CHECKING OVERFLOW.

JSE ICSYM
JMP *50
JMP ENR79

INSERT LABEL.

ACC 2 CONTAINS ADDRESS.

LDA 3,AC379
LDA 0,2,3
STA 0,1,3
LDA 0,2,3
STA 0,2,3
STA 0,2,3
STA 0,3,3
STA 0,3,3
JMP EN379

ENTRY FOUND. SET VALUE FOR LOCTR.

FNC79: LDA 0,1,2
STA 2,AC379
LDA 3,AC379
STA 0,2,3

RESTORE ACC CONTENTS AND RETURN

EN079: LDA 0,AC379
LDA 3,AC379
LDA 0,AC379
LDA 1,AC379
LDA 2,AC379
LDA 3,AC379
JMP 0,5

ERROR OVERFLOW IN TABLE.

FNC75: KX: /<11>1<15>OVERFLOW, ABORT/
MEANS: LDA 0,LEAPS
       STA 1,LEAPS

NAME: ORF77.

FUNCTION: PERFORMS THE LOGICAL OR OPERATION ON INPUTS.

INPUT: LOCATIONS #10, #12, #AC 3 + 1#.

OUTPUT: LOCATION #AC 3 + 2#.

CALLS: NONE.

CALLED-BY: STB79, EBR26, SER 14.

GLOBAL-VARIABLES-USED: NONE.

GLOBAL-VARIABLES-CHANGED: NONE.

ERROR-BITS-SET: NONE.

LOCATIONS FOR SAVING ACC CONTENTS:

AC177: #BLK 1
AC177: #BLK 1
AC177: #BLK 1
AC177: #BLK 1

SAVE ACC CONTENTS:

STA 0,AC077
STA 1,AC177
STA 2,AC277
STA 3,AC377

LOAD IN AND PERFORM OR FUNCTION.
LDA 1,1,3
LDA 1,1,3
LDA 1,1
LDA 1,1
AND 1,1
STA 1,2,3

RESTORE ACC AND RETURN.
LDA 0,AC0??
LDA 1,AC1??
LDA 2,AC2??
LDA 3,AC3??
JNP 3,3

NAME.
FUNCTION.
INPUT.
OUTPUT.
CALLS.
CALLED-BY.
GLOBAL-VARIABLES-USED.
GLOBAL-VARIABLES-CHANGED.
ERROR-BITS-SET.

SPACE FOR SAVING ACC CONTENTS.
AC1111, ,BLK 1
AC1111, ,BLK 1
CH111: 1

SAVE ACC CONTENTS.

UP111: STA J,AC111
       STA 1,AC111

UPDATE COUNTERS.

LD A J,LOCTR
ACC 1,CH111
STA 0,LOCTR

LD A J,LOCTR
ACC 1,LENGTH
STA 0,LOCTR

RESTORE ACC CONTENTS AND RETURN.

LD A J,AC111
LD A 1,AC111
JMP 0,3

NAME.

FUNCTION.

ENTER PASS1 INTO ERROR TABLE.

INPUT.

NONE.

OUTPUT.

NONE.

CALLS.

NONE.

CALLED- BY.

PASS1, PASS2.

GLOBAL-VARIABLES-USED.

PASS1, EDIT1, EDIT3, PFLG.

GLOBAL-VARIABLES-CHANGES.

...
EFFECT - BITS-SET.

SPACE FOR SAVING ACC CONTENTS.

ACF1:  JLK  1
ACF2:  JLK  1
ACF3:  JLK  1
ACF4:  JLK  1

CONSTANTS.

CN06:  62
CN109:  1

SAVE ACC CONTENTS, ENTRY.

EN110:  STA  0,ACF1
EN111:  STA  1,ACF2
EN112:  STA  2,ACF3
EN113:  STA  3,ACF4

CHECK WHETHER PREV. IS ZERO. IF SO, RETURN.

LDA  0,PREV
JCV  1,SHR
JMP  EN309

CHECK WHICH PASS AND SET ACC 1, ACC 2 TO
EPS1  OR EPS2  AND L1EP  OR  EZEPT  ACCORDINGLY.

LDA  0,PSF1
LDA  1,CS1
JMP  PS209

PASS IS 1. SET CORR. VALUES.

LDA  3,EPSTB
JMP  EPS1

PASS IS 2. SET CORR. VALUES.
PSE09: LDA $5,SHFT
LDA $2,SHFT

; CHECK WHETHER TABLE IS FULL. IF SO, RETURN.

PSE09: LDA $4,SHFT
ADV $2
ADV $1
SUBE $1
JMP ENJOY

; TABLE NOT FULL, ENTER ERROR NO.

PSE09: LDA $6,SHFT
STA $2,PEST
STA $1,PEST

; UPDATE LEFT OR E2EPT.

PSE09: INC $1
INC $2

; REPLACE DEPENDING ON PASS.

PSE09: LDA $3,PSFL
LDA $0,PSFL
SUB $0
JMP PSE09

; PASS IS 1. REPLACE AS E1EPT.

PSE09: STA $2,E1EPT
JMP ENJOY

; PASS IS 2. REPLACE AS E2EPT.

PSE09: STA $2,E2EPT

; RESTORE ALL CONTENTS AND RETURN.

PSE09: LDA $1,ACU09
LDA $2,ACU09
JMP ACU09

;
NAME: GL941
FUNCTION: GET CHAR TO THE FIRST BLANK CHARACTER IN IISN AFTER THE CURRENT VALUE.

INPUT: none.
OUTPUT: CHTR POINTS TO BLANK CH.

CALLS: none.

CALLED BY: none.

GLOBAL-VARIABLES-USED: none.
GLOBAL-VARIABLES-CHANGED: none.
ERROR-BITS-SET: none.

SPACE FOR SAVING ACC
SPACE FOR SAVING ACC CONTENTS:

AC141: 1, JX 1
AC141: 1, JX 1
AC341: 1, JX 1

CONSTANTS:

GL41: 4, JX
GL41: SAVE ACC CONTENTS, ENTRY POINT.

GL41: STA 1, GL41
GL41: STA 2, AC141
GL41: STA 3, AC341

GET NEXT WORD:

GL41: JUS GL941
; CHECK WHETHER BLANK OR NOT.
GL41: 1, GL41
; BLANK.
SUBE  UNSIG
JMP  FAD341

CH. IS NOT BLANK, SET NEXT CHARACTER.
JMP  CHPTO
JMP  HXT41

SEPARATOR FOUND, RESTORE ACC CONTENTS AND RETURN.
FAD41:  LOA  J,ACJ41
        LOA  1,AC141
        LOA  2,AC241
        JMP  AC341

NAME.  COPBA
FUNCTION.  SETS CHPTO TO NEXT NON-BLANK CHARACTER.
INPUT.  NONE.
OUTPUT.  CHPTO POINTING TO NON-BLANK CHARACTER.
          TAKES Z ACC 3 + RETURN
          IF ERROR.
          TAKES Z ACC 3 + 12 RETURN
          OTHERWISE.
CALLS.  EME26; ICHK1.
CALLED-BY.  LOTS.
GLOBAL-VARIABLES-USED.  NONE.
GLOBAL-VARIABLES-CHANGED.  CHPTA.
ERROR-BITS-SET.  BIT NO. 16 (F).

SPACE FOR SAVING ACC CONTENTS.
ACCS2:  .BLK 1
AC123:  BLK  1  
AC223:  BLK  1  
AC423:  BLK  1  

CONSTANTS.

CH.241:  49  
CH.223:  123  
CT.241:  BLK  1  
TE.241:  12  

SAVE ACC CONTENTS.

ST.123: STA  0,AC.243  
        STA  1,AC.243  
        STA  2,AC.243  
        STA  3,AC.243  

SET COUNTER TO COUNT UPTO END OF LINE.

LJIA  L,CHPTC  
LJIA  1,CH.223  
SUIE  0,1  
LJIA  1,1  
ST.1,ST.223  

GET NEXT CH. AND CHECK.

NXT23: JERS  GCH.631  

CHECK OUT CHARACTER.

LJIA  1,ST.223  
SUIE  1,CH.223  
JMP  BLK.223  
JMP  FND.223  

CH. IS A BLANK, GET NEXT CH.

BLK.223: ISZ  CHPTC  
         JSZ  CTR.223  
         JMP  NXT23  

ENTER GCODE BIT NO. 16.
(ENO NON-BLANK CH. FOUND).
NAME:  
FUNCTION:  PROCESS DP-CODE, OPERANDS.
           IN PASS 1, SET LENGTH.
           IN PASS 2, SET LENGTH
           AND SET VSPACE (ASSEMBLE WD).
INPUT:   NONE.  
OUTPUT:  NONE.  
CALLS:   SCP23, IR24, SFM25.
CALLED-BY: PASS-1, PASS-2.
GLOBAL-VARIABLES-CHANGED: A3FLG.
ERROR-BITS-SET: GMBEG.
SPACE FOR SAVING ACC CONTENTS:

    A6C071   JSK  1
    A6C074   JSK  1
    A6C071   JSK  1
    A6C071   JSK  1
    A6C071   JSK  1
SAVE ACC CONTENTS, ENTRY POINT.

1071 STA 3, AC207
STA 2, AC207
STA 1, AC207

SET CHPT.

JMP SCP23
JMP EN057

LINE IS OKAY, CHECK VALID TERMINATION.

JMS VTR24

CHECK AFGG TO SEE IF LINE IS QUESTIONABLE.

LDA 3, AGFLG
JMP EN057

LINE IS OKAY, SEARCH TABLES AND SERVICE.

JMS SP125

SET OM3EG AND RESTORE ACCS, RETURN.

EN057 LDA 3, CHPT
STA 3, OME3EG
LDA 2, AC207
LDA 1, AC207
JMP AC207

NAME.

FUNCTION.

SETS CHARACTER POINTED AT BY CHPT FROM INGIN.

INPUT.

NONE.

OUTPUT.

IN ACC U PRIGHT ADJUSTED.

118
CALLS: NONE.
CALLET-BY: LOTS.
GLOBAL-VARIABLES-USED: CHPT, INSLN.
GLOBAL-VARIABLES-CHANGED: NONE.
ERROR-BITS-SET: NONE.

SPACE FOR SAVING ACC CONTENTS:

CTE-11: BLK 1
CTE-21: BLK 1
CTE-31: BLK 1

CONSTANTS:

MEK-11: 377
CH6-11: 121
CHX-11: 40

ENTRY POINT, SAVE ACC CONTENTS:

SC11: DiTh 1, ACl61
STH 2, 30321
STA 3, 3AC381

CHECK CHPT GREATER THAN 11

LJA 0, CHPTL
LJA 1, CHK61
SUBS 1, 3CHNO
JMP ERR-11

FIND OUT WHICH WORD AND WHICH HALF:

LJA 0, CHPTL
LJA 1, CH151
SUB 1, J
JSZ 1, J, SZC

CARRY IS NON-ZERO, SECOND HALF TO BE LOADED.
JMP L2001

CARRY IS ZERO, FIRST IS NCC.
ACC J CONTAINS THE NO TO BE LOADED.

LDA 2, IN CLN
ADC 0, 2
LDA 0, 2
JMP 1, HSKL
LDA 1, HSKL
AND 1, 0

ACC J CONTAINS FEED, NO.

JMP EN361

CARRY IS NON-ZERO, SECOND HALF TO BE LOADED. ACC J CONTAINS WORD TO BE LOADED.

JMP 1:
LDA 2, IN CLN
ADC 0, 2
LDA 0, 2
LDA 1, HSKL
AND 1, 0
JMP EN361

CMP A, GREATER THAN 32.
SEND BACK BLANK.
SET CHPTR TO 31.

LDA 0, 32
STA 0, CHPTR
LDA 0, 32

ACC J CONTAINS FEED, NO.

JMP 1:
LDA 1, CHPTR
LDA 2, AC181
JMP 2, AC181
INPUT.

OUTPUT.

CALLS.

CALLED-BY.

GLOBAL-VARIABLES-USED.

GLOBAL-VARIABLES-CHANGED.

ERROR-BITS-SET.

SPACE FOR SAVING ACC CONTENTS.

AC124:   1: LX 1
AC124:   1: LX 1
AC224:   1: LX 1
AC324:   1: LX 1

CONSTANTS.

BLK24:   4: J ; BLANK
CH24:   101
CH324:   102
CN124:   123
CN324:   1
CN424:   15
THF24:   3

SAVE ACC CONTENTS.

VT124:   STA   0,AC024

TERMINATOR: I.E. CHECK IF I'S 1 F, E E 7.

CUPRIN: STARTING TO FIRST CHARACTER IN CP-DOSE.

ASFLG SET TO 0, 1, 2, OR 3.

U IF INVALID.

1 IF A Z.

2 IF ZA Z.

3 IF A.

CUPRIN POINTS TO FIRST CH. OF CP-DOSE.
LDA 1,CHA24
SUBE 1,1,SHR
JMP CK424

JMP mTF24

CH. IS 1, SET ABFLG TO 1.
CK4241 LDA 1,CHA424 ; DEC 1.
STA 1,ABFLG
JMP CK424

CH. IS 2, SET ABFLG TO 2.
CE241: LOA  L,CH424
STA  J,ABFGLG
JMP  CE241

CHARACTER IS A #, SET ABFGLG TO 1, RETURN.

CH124: JS  J,THP24
STA  .J,ABFGLG
JMP  LE224

CH. IS A OR B, CHECK NEXT CHARACTER.

CH241: LBL  CHKTR
JSR  GC121

ALL J CONTAINS CHARACTER.

LCJ  1,3LK24
SUB  J,11SR
JMP  LE124

INVALID TERMINATOR.
SET BIT 4C, 1(F) OF ERROR WORD.
SET ABFGLG TO 1.

END124: LDA  0,CH524
STA  J,1G124
JSR  ERR26

AG124: JS  J,ABFGLG
STA  0,ABFGLG
JMP  END24

LINE IS QUESTIONABLE.
SET BIT 4C, 1(G) OF ERROR WORD.
SET ABFGLG TO 1.

CHAIN24: STA  J,ABFGLG
LOD  J,CH524
STA  J,ABFGLG
JSR  ERR26

END24: JS  J,ABFGLG
JMP  END24

123
SET CHPT TO BEG OF CHPT.

LDI 244: JSP, CHPT
LDD 24: JSP, CHPT
LDS 2: CHPT

RESTORE ACC CONTENTS AND RETURN.

END 244: LDI 0, ACO 244
LDX 1, ACO 244
LDA 2, ACO 244
JMPS ACO 244

NAME.
FUNCTION.
SEARCH FOR PRODUCT, AND IF FOUND, TRANSFERS TO SERVICE ROUTINE.

INPUT.
CHPT POINTS TO FIRST CH.
IN CP- CODE.

OUTPUT.
LENGTH, CPDC SET.

CALLS.
ENBED, MCH1, BRT4, SIT4, SERVICE ROUTINES.

CALLS- BY.

GLOBAL-VARIABLES- USED.

GLOBAL-VARIABLES-CHANGED.

ERROR- BITS-SET.

SPACE FOR SAVING ACC CONTENTS.

ACI 256: .BLK 1
ACI 256: .BLK 1
ACI 256: .BLK 1
ACI 256: .BLK 1
ACI 256: .BLK 1
CONSTANTS.

CMC25: 4

SPACE FOR SAVOING OPCODE.

CP125: .DLK 1
OP250: .DLK 1
OP125: OP125

SAVE ACC CONTENTS, ENTRY.

SP125: STA 0,AC125
       STA 1,AC125
       STA 2,AC125
       STA 3,AC125

SAVE UPCODE.

CMP25 POINTS TO FIRST CH. SET CHARACTER.

IC=5 GCH61

CHARACTER TO IN AC 0.

MVR 6,0,1
INC CH-1
JZ 6 GCH61
AND 0,1
STA 1,CP125 ; FIRST NO.
JZ 6 CH-1
MVR 6,0,1
STA 2,CP125 ; SECOND NO.

SET CMP25 TO POINT TO END OF CH-CODE.

IF 2 CH-1

CALL CH ROUTINE TO SEARCH FOR.

LDA 0,OPA25
STA 0,AC125
JSPK SPT15

CP125: .DLK 1
AG25: .DLK 1
LDA 0,AG225
TEST WHETHER PRESENT IN NOT.
    LDA 0, J, 0, 0, 0, 0, 0, 0
    STA J, 0, 0, 0, 0, 0, 0, 0
    JSEK SMU4

A57261: BKK 1
A64251: 0

TEST WHETHER PRESENT IN NOT.
    LDA 0, J, 0, 0, 0, 0, 0, 0
    IEC 0, 0, 0, 0, 0, 0, 0, 0
    FSEL IS NON ZERO,_OPCODE IS PRESENT.
    JRP PCI25

ERRAY, NOT PRESENT IN EITHER TABLE.
    ENTER ERROR BIT NO. 4
    STA 0, 0, J, 0, 0, 0, 0, 0
    JSEK EEN425
A65251: BKK 1
A7325 JRP ENJ25

ACU CONTAINS SERVICE ROUTINE ADDRESS.
FSE251: DKA 0, 0, 0
FSE25 JSA 0, 0, 0, 0

RESTORE ACU CONTENTS AND RETURN.
ENJ251: LDA 0, 0, 0, 0, 0, 0, 0, 0
    LDA J, 0, 0, 0, 0, 0, 0, 0
    JSA J, 0, 0, 0, 0, 0, 0, 0
    JNPS AC325
NAME: SP43
FUNCTION: SEARCHES THE P.O.T.
INPUT: STARTING ADDRESS OF LOCAL SPACE WHERE OP-CODE IS STORED (IN A.C. 3 + 2K).
OUTPUT: SERVICE ROUTINE ADDRESS IF PRESENT; OTHERWISE (IN A.C. 3 + 1), RETURNS TO ADD. 3 + 2K.

CALLS: NONE.
CALLED-BY: SP25.
GLOBAL-VARIABLES-USED: none.
GLOBAL-VARIABLES-CHANGED: none.
ERROR-BITS-SET: none.

SPACE FOR SAVING A.C. CONTENTS:

ACC 431: 
AC 143: 
AC 243: 
AC 343: 

CONSTANT: 0

TEMPORARY LOCATIONS:

POP 431: 
PO 431: 

SAVE ACC CONTENTS, ENTRY POINT:

SP 431: STA 3, ACC 43
SEARCH TABLE FOR MATCH.
STORE PORTS, PORT IN TEMPORARY LOCATIONS.

LD A 6, POEPT
STA 0, POEPT
STA 0, POPTR

LOR 43:
LD A 5, POEPT
LOAD 2, POPTR
SUB A 0, 2
def;
IMP NF 43

WHOLE TABLE HAS BEEN SEARCHED.

COMPARE.

LD A 0, 3
LD A 3, 2
SUB A 3, 2
IMP NF 43

NO MATCH

FIRST NO MATCHES TRY SECOND HD.

LD A 0, 3
LD A 3, 2
SUB A 3, 2
IMP NF 43

NO MATCH

MATCH FOUND, LOAD SERVICE ROUTINE ADDRESS.

LD A 0, 3
STA 0, 3
IMP 3, 43

NO MATCH, TRY NEXT ENTRY.
NOT FOUND, SET RETURNING VALUE TO 1.

RESET ACC CONTENTS AND RETURN.

NAME.
FUNCTION.
PRINT OUT THE OUTPUT LINE AFTER DELETING TRAILING BLANKS.
INPUT.
OUTPUT.
one.
calls.
one.
called-by.
put75.
global-variables-used.
putc, errp, pageu.
global-variables-changed.
scfls, slo18.
error-bits-set.
one.
space for acc contents.
ac1821: 1, slk 1
ac1821: 1, slk 1
AC2621: 3LX 1
AC2721: 3LX 1

* CONSTANTS AND COUNTERS.
C1: 21: 74
C2: 21: 1
C3: 21: 20143
C4: 21: 1

* SAVE ACC CONTENTS ENTRY.
P=1:2: STA 1,AC062
STA 1,AC162
STA 1,AC262
STA 1,AC362

* CHECK IF NECO.
LD A 4,BCFLG
LD B J4,4CHR
JMP EN762

* SET DEVICE CODE.
LD A J,SLDES
STA J,OUTDV

* PRINT OUT DF,LF.
LD A J,CH232
LD A PL178
LD A J,CH232
LD A PL178

* SET DF COUNTER TO COUNT UP TO 60 WORDS.
LD A 4,21352
STA 4,STR32

* PRINT OUT.
LD A 2,346TLL
LD A J,CHUB2
ADD 0,2
<table>
<thead>
<tr>
<th>NAME</th>
<th>SMT44</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTION</td>
<td>SEARCHED H.C.</td>
</tr>
<tr>
<td>INPUT</td>
<td>STARTING ADDRESS OF LOCAL</td>
</tr>
<tr>
<td></td>
<td>SPACES WHERE CODE TO</td>
</tr>
<tr>
<td></td>
<td>STORED (IN ACC. + 3 + 2).</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>SERVICE ROUTINE ADDRESSES IF</td>
</tr>
<tr>
<td></td>
<td>PRESENT, OTHERWISE (IN ACC.</td>
</tr>
<tr>
<td></td>
<td>ACC 3 + 1).</td>
</tr>
<tr>
<td></td>
<td>RETURN TO ADDRESS ACC 3 + 2</td>
</tr>
<tr>
<td>CALLS</td>
<td>NONE.</td>
</tr>
<tr>
<td>CALLED BY</td>
<td>SP+25.</td>
</tr>
<tr>
<td>GLOBAL VARIABLES</td>
<td>REACT, OPT.</td>
</tr>
<tr>
<td>CHANGED</td>
<td>ISVAL.</td>
</tr>
<tr>
<td>ERROR BITS SET</td>
<td>NONE.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>CONSTANTS/COUNTERS</td>
<td></td>
</tr>
<tr>
<td>ACC63</td>
<td>1</td>
</tr>
<tr>
<td>BLK44</td>
<td>1</td>
</tr>
<tr>
<td>UPP44</td>
<td>1</td>
</tr>
<tr>
<td>IND44</td>
<td>1</td>
</tr>
<tr>
<td>MOP44</td>
<td>1</td>
</tr>
<tr>
<td>MCP44</td>
<td>1</td>
</tr>
<tr>
<td>OME44</td>
<td>1</td>
</tr>
<tr>
<td>CH44</td>
<td>1</td>
</tr>
<tr>
<td>O1.144</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>SPACE FOR SAVING ACC CONTENTS.</td>
</tr>
<tr>
<td>ACC144</td>
<td>1</td>
</tr>
<tr>
<td>BLK144</td>
<td>1</td>
</tr>
<tr>
<td>ACC244</td>
<td>1</td>
</tr>
<tr>
<td>BLK244</td>
<td>1</td>
</tr>
<tr>
<td>ACC344</td>
<td>1</td>
</tr>
</tbody>
</table>
SAVE ALL CONTENTS, ENTRY POINT

LWR44  STA    0, LWR44
        STA    1, LWR44
        STA    2, LWR44
        STA    3, LWR44

INITIALIZE LWR44, UP44, KEY44, IND44, TOP44

LDA    0, J3
LDA    0, KEY44
LDA    0, UP44
STA    0, LWR44
STA    0, LWR44
STA    0, LWR44
STA    0, LWR44
STA    0, LWR44

CHECK IF UP44 IS LESS THAN LWR44.
IF SO, SEARCH HAS INDEED UNSUCCESSFULLY.

L=244 LDA    0, LWR44
LDA    1, UP44
SLEEP J=1, SLC
JMP    UP44 : UNSUCCESSFUL

SET INDEX (IND44) TO LOWER BOUND OF (LWR44+UP44)/2.

INC J=1, J
STA    0, IND44

LOAD ADDRESS OF ENTRY CURR. TO INDEX.

LDA    0, IN144
SUB    J=1
ADD    J=1
STA    3, UP44

AUX 1, 3
LDA    2, KEY44
LDA J, J, J, J
LDA 1, 1, 1, 1
JMP NZ6544

CHECK SECOND NO.

LDA J, J, J, J
LDA 1, 1, 1, 1
JMP NZ6544

NO MATCH, GET NEXT ENTRY TO BE CHECKED.
ALL CONTAIN MISMATCHED NO. OF KEY.

CHECK WHETHER K LESS THAN K(I).

LDA J, J, J, J
JMP LE544

K LESS THAN K(I), SET UP44 = INC J - 1

LDA J, J, J, J
SUB 1, J
STA J, J, J44
JMP LE244

K GREATER THAN K(I), SET LUP44 = INC J + 1

LE544:
LDA J, J, J, J
INC J, J
STA J, J, J
JMP LE244

MATCH NOT FOUND, SET RETURNING VALUE TO J.

NFO44:
LDA J, J, J, J
LDA 3, 1, 1, 1
STA J, J, J
ENTRY FOUR, AC 3 CONTAINS AC 1, AC 2, AC 4 MATCHING ENTRY.
LEFT HALF OF VALUE CONTAINS FORMATING NO.
RIGHT HALF CONTAINS BASE VALUE.

ISOLATE BASE VALUE.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LOA</td>
<td>0, 2, 3</td>
<td>: LOAD VALUE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOA</td>
<td>1, 16K44</td>
<td>: LOAD MASK</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STA</td>
<td>0, 35VAL</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ISOLATE FORMAT NO.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LOA</td>
<td>0, 3, 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOA</td>
<td>1, 16K44</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACS</td>
<td>0, 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AND</td>
<td>1, 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

LOAD SERVICE ROUTINE ADDRESS.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LOA</td>
<td>3, 3ER44</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AOS</td>
<td>0, 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOA</td>
<td>1, 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SET RETURNING VALUE.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LOA</td>
<td>3, AG344</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STA</td>
<td>1, 1, 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RESTORE ACC CONTENTS AND RETURN.

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>LOA</td>
<td>3, AG444</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOA</td>
<td>1, AG144</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOA</td>
<td>2, AG244</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LOA</td>
<td>3, AG344</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>JMP</td>
<td>2, 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

NAME.

FUNCTION.

PRINT OUT LISTING.
INPUT:                      NONE.
OUTPUT:                    NONE.
CALLS:                     CALL 30,LOC 30,LOC 30,LOC 30,LOC 30,LOC 30,LOC 30,LOC 30,LOC 30.
CALLED BY:                 PASS 1,PASS 2.
GLOBAL-VARIABLES-USED:    LTF 1,PRE 1.
GLOBAL-VARIABLES-CHANGED: NONE.
ERROR-BITS-SET:           NONE.

SPACE FOR SAVING ACC CONTENTS:
ACC 110:                   BLK 1
ACC 210:                   BLK 1
ACC 310:                   BLK 1

SAVE ACC CONTENTS, ENTRY POINT:
PS 110:                    STA 0,ACC 110
                         STA 1,ACC 210
                         STA 2,ACC 310

CHECK WHETHER LISTING IS NEEDED:
C3M 1, LTF 1, PRE 1, SHR
JMP LNN 1

LISTING IS NEEDED:
JMP 30, LPL 29
JMP 30, LPL 30
JMP 30, LPL 31
JMP 30, LPL 32
JMP 30, LPL 62

136
JNS  F+34
JMP  END10

CHECK WHETHER ERROR-ENTRY.
LDA  0,3,PRESS
MOV  U,3,CNR
JMP  END10

IT IS A ERROR-ENTRY.
JNSK  ERC26
JNSK  ERC28
JNSK  ERC31
JNSK  PRT32

RESTORE ACUS AND RETURN.
END10:  LDA  J,ACU16
LDA  1,ACU16
LDA  23,ACU16
JMP  ACU10

NAME.  ERC26.
FUNCTION. ENTER LINE COUNTER INTO PARLI AFTER CONVERSION
INTO DECIMAL INTO POSITIONS B-10.

INPUT.  NONE.
OUTPUT.  NONE.

CALLS.  33C46.

CALLED-BY.  POS10.
GLOBAL-VARIABLES-USED.  LRTH.
GLOBAL-VARIABLES-CHANGED.  PRR.
ERROR - BIT - ELSE.

SPACE FOR SAVING ACC CONTENTS.
AC 0001 : BLK 1
AC 0002 : BLK 1
AC 0003 : BLK 1
AC 329 : BLK 1

CONSTANTS.
CAL 291 : 6

SAVE ACC CONTENTS, ENTRY.
EL 129 : STA 0, AC 329
EL 129 : STA 1, AC 329
EL 129 : STA 2, AC 329
EL 129 : STA 3, AC 329

CONVERT INTO DECIMAL AND ENTER.
-2A : LDA 0, LTERM
-2A : LDA 1, D1029
-2A : LDA 1, PRT4
-2A : JNS B0044

RESTORE ACC CONTENTS AND RETURN.
L04 : STA 0, AC 329
L04 : STA 1, AC 329
L04 : STA 2, AC 329
L04 : JMS AC 329

NAME. 24X47.
FUNCTION. CONVERT 16THS INTO 4
HEX CHARACTERS.
INPUT. AC 0.
OUTPUT.

2 words in locations
REG 34, REG 3 + 1.

CALLS.

NONE.

CALLER-3Y.

NONE.

GLOBAL-VARIABLES-USED.

E1031, C1094, RCH17, P1273.

GLOBAL-VARIABLES-CHANGED.

NONE.

ERROR-BITS-SET.

NONE.

SPACE FOR SAVING ACC CONTENTS.

ACC471 3LK 1
ACC47I 3LK 1
ACC47I 3LK 1
ACC471 3LK 1

MCK47: 17

CHARACTER CODES FOR HEX CHARACTERS.

CH47:

60
61
62
63
64
65
66
67
71
71
71
71
101
102
103
105
106

CHR47

SAVE ACC CONTENTS, ENTRY.
A GLUE CONTAINS NO TO BE CONVERTED.

CONVERTED BY SHIFTING, MASKING, AND INDEXING.

<table>
<thead>
<tr>
<th>LDA</th>
<th>1; j, 447</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>1; j</td>
</tr>
<tr>
<td>LDA</td>
<td>1; j, 2</td>
</tr>
<tr>
<td>MOVZ</td>
<td>j, j</td>
</tr>
<tr>
<td>MOVZ</td>
<td>j, j</td>
</tr>
<tr>
<td>LDA</td>
<td>3; j, 447</td>
</tr>
<tr>
<td>AND</td>
<td>2; j, 447</td>
</tr>
<tr>
<td>LDA</td>
<td>3; j, 2</td>
</tr>
<tr>
<td>MOVZ</td>
<td>j, j</td>
</tr>
<tr>
<td>LDA</td>
<td>1; j, 4347</td>
</tr>
<tr>
<td>STA</td>
<td>2; j, 4347</td>
</tr>
<tr>
<td>STA</td>
<td>3; 1, 2</td>
</tr>
</tbody>
</table>

| LDA  | 0; j     |
| LDA  | 0; j     |
| LDA  | 1; j, 447 |
| AND  | 0; 1     |
| LDA  | 2; j, 447 |
| LDA  | 1; j, 2  |
| MOVZ | j, j    |
| MOVZ | j, j    |
| MOVZ | j, j    |


1. STORE ACC AND RETURN.
   
   LOAD 3,AC247
   LOAD 1,AC147
   LOAD 1,AC147
   MOV 2,3

   NAME:
   ESC 32.
   FUNCTION:
   ENTER SOURCE CODE INTO PRTLN FROM 2,140.
   INPUT:
   NONE.
   OUTPUT:
   NONE.
   CALLS:
   P0016.
   CALLED-BY:
   EP-34.
   GLOBAL-VARIABLES-USED:
   INSLN.
   GLOBAL-VARIABLES-CHANGED:
   NONE.
   ERROR-BITS-SET:
   NONE.
   SPACE FOR SAVING ACC CONTENTS:
   AC 121:  0, BLK  1
   AC 122:  0, BLK  1
AC 2
AC 3

CONSTANTS AND COUNTERS.

CH 27 = 5
CH 32 = 33
CSK 21 = 377
CTF 32 = 1

SAVE AND CONTENT ENTRY.

ES1 52:
STA 0, AC2 52
STA 1, AC1 52
STA 2, AC2 52

SET UP COUNTER TO COUNT UPTO 40 HOPS.

LOA 0, CH1 52
STA 0, CT 52

SET RPTVALU.

LOA 0, CH1 52
STA 0, RPT

LOA 2, INSLN

GET NEXT HOP AND INTRODUCE IN RPTLN.

LCP 52:
LOA 0, J, 2
MOV 0, J
LOA 1, MK 52
AND 1, 123
JSRS 2 ERP 03 ; LEFT HALF

ISZ RPT
LOA 0, J, 2
LOA 1, MK 52
AND 1, 123
JSRS 2 ERP 13 ; RIGHT HALF.

UPDATE COUNTERS, INDICES.
CHECK WHETHER ALL HOPS HAVE BEEN ENTERED.
NAME.

FUNCTION.

INPUT.

OUTPUT.

CALLS.

CALLED-BY.

GLOBAL-VARIABLES-USED.

GLOBAL-VARIABLES-CHANGED.

ERROR-BITS-SET.

SPACE FOR SAVING ACC CONTENTS.

AC6371 . BLK 1
AC1631 . BLK 1
AC3631 . BLK 1
AC6631 . BLK 1

SPACE 1 377
MS18311 177400

; SAVE ACC CONTENTS ENTRY.

EEP18311 STA 0, AC.83
      STA 1, AC.163
      STA 2, AC.283
      STA 3, AC.383

; GET THE WORD POINTED TO BY PRPTR.

      LDA 1, PRPTR
      LDA 1, NH.083
      SUB 1, 0
      MOV 0, J, SZE

; ACC 0 CONTAINS WD TO BE LOADED.
; VALUE BETWEEN 0 AND 99.

      JMP NZ083
      ; RIGHT HALF

; CARRY IS ZERO, LEFT HALF TO BE REPLACED.
; ACC 0 CONTAINS THE WD TO BE LOADED.

      LDA 3, PRTRL
      ADD 0, 0
      LDA 0, 1, 1, 3
      LDA 1, MS.083
      AND 1, 0
      ; LEFT HALF.

; AC 0 CONTAINS RIGHT HALF, LEFT HALF IS 0.

      LDA 1, AC.33
      LDA 2, MS.083
      AND 2, 1
      MOV 1, 1

; AC 1 CONTAINS NEXT LEFT HALF, RIGHT HALF IS 0.

; FORM WD.

      ADD 1, 1
      STA 0, 3
      JMP END83

; CARRY IS 'CH-ZERO, RIGHT HALF TO BE REPLACED.
; AC 0 CONTAINS WD TO BE REPLACED.
NZCL3:  LOA  5,PHTLN
        ADD  0,3
        LDA  0,0,0,3
        LDA  1,MS183
        AND  1,0

AC 0 CONTAINS LEFT HALF OF NO, RIGHT HALF IS 0.

LOA  1,AA63
LOA  3,MSK63
AND  2,1

AC 1 CONTAINS RIGHT HALF, LEFT HALF IS 0.

FORM NOC AND REPLACE.

ADD  1,1
STA  0,1,3

RESTORE ACC CONTENTS AND RETURN.
END63:  LOA  1,AC183
        LDA  1,AC183
        LDA  3,AC283
        JMPR  AC383

NAME.
FUNCTION.
ENTER ERROR LETTERS UPTO MAXIMUM OF 4 INTO
PATH FROM 1-4.

INPUT.

OUTPUT.

CALLS.

CALLED-BY.

GLOBAL-VARIABLES-USED.

GLOBAL-VARIABLES-CHANGED.
ERROR-JITS-SET

SPACE FOR SAVING ACC CONTENTS.

AC1221: BLK 1
AC1224: BLK 1
AC2221: BLK 1
AC2224: BLK 1

CONSTANTS.

CH231: 20
CH128: 1
CH228: 4
CHA228: BLK 1
CHA229: BLK 1

CHARACTER CODES FOR ERROR LETTERS.

CHC231: 111
115
114
114
117
1285
123
1265
115
156
1223
1226
1251
1201
1623
163

SAVE ACC CONTENTS ENTRY.

EA128: STA 1,AC228
STA 1,AC1228
STA 3,AC2228
STA 3,AC3228
SET UP COUNTER TO COUNT UNTIL 16, 4

LD A 0, CH 28
STA Z, CH 28
LD A 1, CH 28
STA Y, CH 28

SET PTR VALUE

LD A 0, CH 28
STA P, PTR

CHECK AND INTRODUCE INTO PTR...

LD A 1, PTR
LD A 2, CH 28

LCP 28: INCY 1, LONG
JMP ZLY 28

CARRY IS NOT-ZERO, ENTER LETTER, UPDATE STR.

LD A 3, STR 28
SUB 0, CH 28
ADD 2, 3
LD A 2, CH 28
JMP CH 28
JMP CT 28
JMP END 28

UPDATE COUNTER TO COUNT UNTIL 16.

ZLY 28: JSZ CT 28
JMP LCP 28

RESTORE ACC CONTENTS AND RETURN.

END 28: LD A 0, AC 28
LD A 1, AC 28
LD A 2, AC 28
JMP AC 328
NAME.
FUNCTION.
ENTERS LOCTI INTO PTLH INTO POSITIONS 12-15 AFTER CONVERSION INTO HEX.
INPUT.
OUTPUT.
NONE.
NONE.
CALLS.
NONE.
CALLED-BY.
GLOBAL-VARIABLES-USED.
PTLH, LCFL6, LCOH.
GLOBAL-VARIABLES-CHANGED.
PTLH, LCFH.
ERROR-BITS-SET.
NONE.

SPACE FOR SAVING ACC CONTENTS.
AC0301 . .BLK 1
AC1301 . .BLK 1
AC2301 . .BLK 1
AC3301 . .BLK 1

CONSTANTS.
CL0301 '14 ; DEC 12

SAVE ACC CONTENTS, ENTRY.
LO1301 STA 6,AC030
STA 7,AC130
STA 8,AC230
STA 9,AC330

SET REG VALUE.
SET ACC J VALUE TO LOCTI.
CHECK WHETHER THE ENTERING IS NEEDED.

LDA 0,CHU30
STA 0,PRM3
LDA 0,LOCT4
LDA 1,LOF30
MOV 1,1,323

CALL ON ROUTINE TO CONVERT AND ENTER.

JSRS CVT30

STORE ACC CONTENTS AND RETURN

LDA 0,ACH30
LDA 1,AHL30
LDA 2,ACR30
JSRS AC33

NAME. CVT30
FUNCTION. CONVERT WORD IN ACC 0 INTO HEX C4, AND ENTER THEM INTO PRTN STARTING FROM PRTN.

INPUT. ACC 0.

OUTPUT. none.

CALLS. none.

GLOBAL-3V. none.

GLOBAL-VARIABLES-USED. none.

GLOBAL-VARIABLES-CHANGED. none.

ERROR-BITS-SET. none.

SPACE FOR SAVING ACC CONTENTS.
MKEEN: 377

SAVE ACCESS CONTENT, ENTRY.

CXX164: STA 0, ACC
STA 1, ACC
STA 2, ACC
STA 3, ACC

PEPTA VALUE ALREADY, SET.
ACC J CONTAINS WORD TO BE ENTERED.

CONVERT INTO HEX, ACC J CONTAINS WORD.

JSR ACC
ACC J

ENTER MORE INTO PGTNL, CALL ON PEPTA

LDA 0, AG0
LDS 0, J
LDA 1, MSK8
AND 1, J
JSR5 EPT6
ISZ PEPTR
LDA 1, AG0
LDA 1, MSK8
AND 1, J
JSR5 EPT6
ISZ PEPTR
LDA 0, AG1
MOV 0, J
LDA 1, MSK8
AND 1, J
JSR5 EPT6
ISZ PEPTR
LDA 0, AG1
LDA 1, MSK8
AND 1, J
JSK5  ENTRY

RESTORE ACC CONTENTS AND RETURN.

LDA  J,AC004
LDA  1,AC184
LDA  2,AC254
JMP   AC384

NAME.

FUNCTION.

PUTS OUT A STRING OF
CHARS TERMINATED BY A NULL,
Packed RIGHT TO LEFT.

INPUT.

IN ACC 2, STARTING ADDR.

OUTPUT.

NONE.

CALLEES.

NONE.

CALLED-BY.

PUT75.

GLOBAL-VARIABLES-USED.

LOT3.

GLOBAL-VARIABLES-CHANGED.

NONE.

ERROR-BITS-SET.

NONE.

SPACE FOR SAVING ACC CONTENTS.

AC045:  *BLK   1
AC185:  *BLK   1
AC255:  *BLK   1
AC385:  *BLK   1

WSK5:  177

SAVE ACC CONTENTS, ENTRY POINT.

WFI5:  STA  J,AC004

151
GET NEXT WORKSPACE CONTAINS ADDRESS.

LCPST:

LDA 1, V2
LDA 1, V3
AND 1, V4
JMP PUT77
MOV 1, V5
LDA 0, V6
AND 1, V7
JMP ELP77
INC 2, V8
JMP LCPST

ALL CHARACTERS PRINTED OUT, RETURN.

ENDO:

LDA 1, V9
LDA 1, V10
LDA 2, V11
JMP AC386

NAME:

FUNCTION:

PLOTS OUT CODE QUERY, GETS REPLY AND SETS BNFGL, LTFGL, ENDAD ACCORDINGLY.

INPUT:

NONE.

OUTPUT:

NONE.

CALLS:

N-T36, REC74.

CALLED-BY:

ASC16.

GLOBAL-VARIABLES-USED:

LTOU, KING.

GLOBAL-VARIABLES-CHANGE:

LREU, OUIV, ENDAD.
SPACE FOR SAVING ACC CONTENTS.

<table>
<thead>
<tr>
<th>ACC</th>
<th>BLK</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

MESSG

<table>
<thead>
<tr>
<th>MESSG</th>
<th>MESSG</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

SAVE ACC CONTENTS, ENTRY.

<table>
<thead>
<tr>
<th>M0185</th>
<th>STA</th>
</tr>
</thead>
<tbody>
<tr>
<td>0, AC0 5</td>
<td></td>
</tr>
<tr>
<td>1, AC1 6</td>
<td></td>
</tr>
<tr>
<td>2, AC2 6</td>
<td></td>
</tr>
<tr>
<td>3, AC3 6</td>
<td></td>
</tr>
<tr>
<td>6, JIND</td>
<td></td>
</tr>
<tr>
<td>7, JIND</td>
<td></td>
</tr>
<tr>
<td>9, MEO A</td>
<td></td>
</tr>
<tr>
<td>0, JUTO A</td>
<td></td>
</tr>
</tbody>
</table>

PRINT OUT QUERY.

<table>
<thead>
<tr>
<th>M185</th>
<th>LOA</th>
</tr>
</thead>
<tbody>
<tr>
<td>2, MESSG</td>
<td></td>
</tr>
</tbody>
</table>

SET REPLY.

| M185 | RED74 |

REPLY IS IN ACC 6.

CHECK WHAT REPLY IS AND TAKE ACTION.

<table>
<thead>
<tr>
<th>M185</th>
<th>SUB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, M185</td>
<td></td>
</tr>
<tr>
<td>1, JSNR</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M185</th>
<th>MALT</th>
</tr>
</thead>
<tbody>
<tr>
<td>60 ORTAL</td>
<td></td>
</tr>
</tbody>
</table>

153
; MODE IS 1
LDA 1,CH3E
STA 1,CH4E
JMP MCH4E
JMP MCH5E
JMP ACH5E
JMP 1,CH5E
; MODE IS 2
; MODE IS 3
; REPEAT INDIVIDUALLY.

; MODE = 1, SET ENDADE TO 0, RETURN.
MCH3E: LDA 0,12
STA 0,ENDADE
JMP END8E

; MODE = 2, SET ENDADE TO 1, LTFLG TO J, RFLG TO 1.
MCH5E: LDA 1,STDA
STA 0,LTFLG
LDA 0,STDA
STA 0,ENDADE
STA 0,ENDADE
JMP END8E

; MODE IS 3, SET ENDADE TO 1, LTFLG TO J, RFLG TO 0.
MCH8E: LDA 1,STDA
STA 0,LTFLG
LDA 0,STDA
STA 0,ENDADE
STA 0,ENDADE
JMP END8E

; PEFILT OUT LF AND CF.
; RESTORE ACC CONTENT AND RETURN.
END8E: LDA 0,10T3E
LDA 0,10T4E
LDA 0,10T5E
LDA 0,10T6E
LDA 0,10T7E
JMP ENDS

;
CHECK LDF G TO SEE IF ERROR.
LLL

LDA 3, 35LC
LOY 3, JOY
IMPS 3, P52

SET BIT NO. 12(P).

LDA 6, THL52
STA 0, A652
LDS 0, E626

ARGS21: BELK 1

RESTORE ACC CONTENTS AND RETURN.

STP52: LDA 1, AC352
       LDA 1, AC152
       LDA 2, AC252
       IMPS AC352

NAME.

FUNCTION.

INPUT.

OUTPUT.

CALLS.

Called-by.

GLOBAL-VARIABLES-Jul.d.

GLOBAL-VARIABLES-changed.

ERROR-BITS-SET.

SPACE FOR SAVING ACC CONTENTS.

AC71: .BLK 1
CH171: 1
SAVE ACO CONTENTS AND ENTRY POINT.

LDA #CH171
STA J,ACO

SET LENGTH TO 1

LDA J,LENGTH
STA J,LEN

SET SPACE.

LDA J,BSPACE
STA J,U

RESTORE ACO CONTENTS, RETURN.

LDA J,ACO
JMP U,
SPACE FOR SAVING ACO CONTENTS.

AC115:  .BLK  1
AC115:  .BLK  1
AC215:  .BLK  1
AC315:  .BLK  1

CH115:  1
ENTRY POINT, SAVE ACO CONTENTS.

EI115:  STA   6,AC115
        STA   1,AC115
        STA   2,AC215
        STA   3,AC315

CHECK ENFLG.

LDA   0,ENFLG
LDA   1,CH115
SUBE   0,1,325
JMP   EN335

CHECK IF LOCAL IS 0.

LDA   0,LOCAL
JMP   EE336

CHECK PAFLG.

LDA   0,PAFLG
LDA   1,CH115
SUBE   0,1,212
JMP   PL315

CALL ON ROUTINE TO ENTER LHDN INTO BOUT.
CHECK LOCAL AGAINST NCHDS.

JMP<   EE36
LDA   0,NCHDS
LDA   1,LHDN
SUBE   0,1,NHD
PC115:  JMP<   PCH37
NAME: AC15

FUNCTION: INITIALISE INPUT, OUTPUT, HOME BUFFER.

INPUT: CALLS: GLOBAL-VARIABLES-USED: ERROR-BITS-SET:

OUTPUT: CALLED-BY: GLOBAL-VARIABLES-CHANGED: LOCAL, GLOBAL:

CALLS: LOCAL, GLOBAL:

GLOBAL-VARIABLES-USED: OUT, LOOP.

GLOBAL-VARIABLES-CHANGED: LOCAL, GLOBAL.

ERROR-BITS-SET: ONE.

SPACE FOR SAVING ACC CONTENTS:

AC15: ;BLK 1
AC415: ;BLK 1
AC2415: ;BLK 1
AC3415: ;BLK 1

CONSTANTS:

C0: 0
C1: 0
C2: 0
C3: 0
C4: 0
CTF 35: .BLK 1
M3K 35: 377

; SAVE ACC CONTENTS, ENTRY.
IE1351: STA 0, AC 36
STA 1, AC 36
STA 2, AC 36
STA 3, AC 36

; INITIALISE EBCUT.
LD 4, 1356
STA 3, CT 36
;
LD 2, 310077
LD 0, G1 36
LB1351: STA 1, I, 2
JMP 31036
SET 51.
LD 6, 1356
LDA 2, 310077
STA 1, I, 2

; SET ADDRESS VALUE.
LDA 0, LOCTR
MOV 0, 0
LDA 1, MSK 35
AND 1, I
STA 0, 2, 2
LDA 0, LOCTR
AND 1, I
STA 0, 3, 2

; SET LOCAL TO 4.
LD 0, CN 35
STA 0, LOCAL

; RESTORE ACC AND RETURN.
LOAD 0, AC336
LOAD 1, AC336
LOAD 2, AC336
JMP $ AC336

FUNCTION.
ENTERS LENGTH BYTES FROM MESSAGE INTO OUTPUT.
UPDATES LOCAL.

INPUT.
NONE.

OUTPUT.
NONE.

CALLS.
NONE.

CALLED-BY.
NONE.

GLOBAL-VARIABLES-USED.
OUT, LENGTH.

GLOBAL-VARIABLES-CHANGED.
LOCAL, OUT.

ERROR-FLAG-SET.
NONE.

SPACE FOR SAVING ACC CONTENTS.
ACC336.
BLK 1
AC136.
BLK 1
AC236.
BLK 1
AC336.
BLK 1
CTF361.
BLK 1
MSK361.
377

SAVE ACC CONTENTS, ENTRY.
ES136.
STA 0, AC036
STA 1, AC136
STA 2, AC236
STA 3, AC336

ENTER LENGTH BYTES INTO OUT.
NAME. 

FUNCTION. GETS CHARACTERS, ENTERS IT INTO GOUT AND UPDATES LOCAL.

INPUT. NONE.

OUTPUT. NONE.

CALLS. NONE.

CALLED BY. PCH37.

GLOBAL-VARIABLES-USED. GOUT.

GLOBAL-VARIABLES-CHANGED. GOUT, LOCAL.

ERROR-BITS-SET. NONE.

SPACE FOR SAVING ADD CONTENTS.

AC655: *BLK 1
AC1650: *BLK 1
AC665: *BLK 1
AC1650: *BLK 1
CT1650: *BLK 1
MSK550: *BLK 377

SAVE ADD CONTENTS, ENTRY

SU1651 STA 0,AC150
STA 1,AC150
STA 2,AC150
STA 3,AC150
STA 4,LOCAL
STA 0,CT150
JMP END550

JUZ 0, J
JSZ 0, CT150
JAP +2
JMP  END50

LDA  2,3
LDA  2,2
LDA  1,J,2
JSZ  C+50
JMP  END50

LBD50:  LDA  2,2
LDA  1,J,2
JSZ  C+50
JMP  LBD50

ACC O CONTAINS THE SUN BACK OUT AND COMPLIMENT FOR CHECKSUM.

END50:  LDA  1,15K50
STA  1,1
STA  0,1,2
ISZ  LOCAL

RESTORE ACC CONTENTS AND RETURN.

LDA  1,AD150
LDA  1,AD150
LDA  2,AD350
JAPS  AD350

NAME:    PCH 47
FUNCTION: FLUSHES THE OBJECT CODE BUFFER, BLDOUT.
          GETS CHECKSUM, CONVERTS INTO HEX, AND PUNCHES OUT.
INPUT:    NONE.
OUTPUT:   NONE.
CALLS:    SUN 53, PUN 7, BA8 47.
CALLCC-BY.

GLOBAL-VARIABLES-USED.

GLOBAL-VARIABLES-CHANGED.

ERROR-BITS-SET.

SPACE FOR SAVING ACC CONTENTS.

ACC 371  .BLK 1
AC 1371  .BLK 1
AC 3271  .BLK 1
AC 371  .BLK 1

CONSTANTS

CN 371  0
CN 371  0
CT 371  .BLK 4
FIV 371

STORE ACC CONTENTS, ENTRY.

PC 1371
STA 0, ACC 371
STA 1, AC 1371
STA 2, AC 2371
STA 3, AC 371

GET CHECKSUM

INTRODUCE BYTE COUNT.

LDA 2, [N 03]
LDA 0, [LOCAL]
INC 0, 1
LDA 1, [CN 237]
SUB 1, 0
STA 0, 1, 2
JMP 3, [15]

CONVERT BINOUT

LDA 0, [LOCAL]
```
LB37:  LDA   J, J, 2
       BNE   237
237:   STA   J, AC237
       JNZ   237
JMP   LBL37

ALL M'S ARE CONVERTED, PUNCH OUT
JSE   PNL12
JSE   PNL40
JNE   PNL12

SET LOCAL TO 3.
L37:   LDA   0, CH, 37
       STA   0, LOCAL

RESTORE ACC CONTENTS AND RETURN.
LDA   0, AC37
LDA   1, AC137
LDA   2, AC237
JMP   AC337

NAME.     FUMAS.
FUNCTION.  PUNCH OUT ALL WORDS 
           IN 3OUT FROM 1 TO LOCAL.
INPUT.     NONE.
OUTPUT.    NONE.
```
CALLS.
CALLED-3Y.
GLOBAL-VARIABLES-USED.
GLOBAL-VARIABLES-CHANGED.
ERROR-HITS-SET.

SPACE FOR SAVING ACC CONTENTS,
AC0401  .SLK  1
AC1401  .SLK  1
AC2401  .SLK  1
AC7401  .SLK  1

CONSTANTS.
CF4401  .SLK  1
MSK401  .SLK  177

SAVE ACC CONTENTS.
PU1401  STA  3,AC343
        STA  3,AC414
        STA  3,AC241
        STA  3,AC341

PUNCH OUT HLLS."

SET DEVEICE CODE.
LOD  6,PU000
STA  6,OUT00

SET UP COUNTER.
LOD  6,LOCAL
STA  3,CTR43
LDA 2,3NOUT

PUNCH OUT.

LDA 1;15H43
LDA 0;1,1
AND 1,1
JSRS PUTF75
LDA 0;912
JSRS ALT75
INC 2,2
JSRS CTR46
JMP LEL46

PUNCH OUT NULLS.

JSRS PNL12

RESTORE ACC CONTENTS AND RETURN.

LDA 3,40140
LDA 1,40143
LDA 3,40240
JSRS AL340

---

NAME: GNC67
FUNCTION: GET NEXT CH. FROM INSLN.
INPUT: CURSOR POINTING TO CH. BEFORE READ ON.
OUTPUT: RETURNS THE CH. IN ACC 0.
CALLS: RETURNS A BLANK IF GREATER THAN 0.
CALLED-BY: GNC61
GLOBAL-VARIABLES-USED: LOTS.
GLOBAL-VARIABLES-CHANGES
ERROR-MITS-SET.

SPACE FOR SAVING ACC CONTENTS.
ACC 76: BLK 1
ACC 57: BLK 1
ACC 37: BLK 1

CONSTANTS.
CNSE76: 121
BLK 57: 40

ENTRY POINT, SAVE ACCS.
CN 157: STA 2,46287
CN 157: STA 1,46187
CN 157: STA 3,46287

INC CHPT & CHECK.
ISZ CHPT
LDA 0,0 CHPT
LDA 3, CN 537
SUBL 3, HNC
JMP ERRE7

CHPTR OKAY, SET CH.
JSR 5 CN 61

AC J CONTAINS CH RETURN.
JMP END;

CHPTR IS 31.
EFF 57: LDA 0, BLK 87
STA 4, CHPT

RESTORE ACC RETURN.
ENGLISH

FUNCTION:
CONVERTS A SERIES OF ASCII
CHAR. CH. INTO A BINARY NO.

INPUT:
CHPTR POINTS TO CH.
BEFORE STRIPS.
CH. ARE REQUESTED THRO
A CALL TO GHOST.

OUTPUT:
CHPTR POINTS AT 0000.
ACC 0 CONTAINS 0000.
ACC 1 CONTAINS VALUE.

CALLS:
GHOST.

CALLED-BY:
NONE.

GLOBAL-VARIABLES-USED:
NONE.

GLOBAL-VARIABLES-CHANGED:
NONE.

ERROR-BITS-SET:
NONE.

SPACE FOR SAVING ACC CONTENTS.

ACC931  ,BLK  1
ACC931  ,BLK  1

CONSTANTS:
CH931   61    :CH 1
CH7631  60    :CH 0
EM931   ,BLK  1

SAVE ACC,ENTRY,INITIALIZE RUNNING DAY.
BN1931  STA  2,AC293
ST A 0, 0, 0, 0
STA 0, 0, 0, 0

GET NEXT CH AND UPDATE RUNNING SUM

LBL93:  J M S  G N 0 9 3
        L D A 1, 1 1 0 H
        L D A 3, 1 1 0 H
        R N Z E 1, 1 0 H
        R N Z E 1, 1 0 H
        J M P B A K 9 3

        S U B 1, 1 1 0 H
        L D A 1, 1 0 9 3

MULTIPLY BY 2 AND ADD

S U V Z L 1, 1 1 0 H
        A D D 1, 1 1 0 H
        S T A 1, 1 0 N 9 3
        J M P L E L 9 3

A C C U CONTAINS BREAK CH, RETURN.

S E X 9 3:  L D A 1, 0 N 9 3
        L D A 3, 1 A C 2 9 3
        J M P S A L 9 3

NAME:  HEX94
FUNCTION:  CONVERTS A SERIES OF ASCII
HEX CH's INTO A BINARY NO.
INPUT:  CH:  POINTS TO CH.
BEFORE STRING, CH: ARE REQUESTED THEN
A CALL TO G N 0 9 3.

OUTPUT:  DUP:  POINTS AT BREAK.
AD D U CONTAINS BREAK.
CALLS

CALLED BY

GLOBAL VARIABLES USED

GLOBAL VARIABLES CHANGED

ERROR BITS SET

SPACE FOR SAVING ACC CONTENTS

AC894: BLK 1

AC994: BLK 1

CONSTANTS

CHS94: 67

CHZ94: 60

CHF94: 71

CHI94: 101

CHF54: 106

CHM94: BLK 1

SAVE ACC, ENTRY, INITIALISE DBLE

HE194: STA 2, AC294

STA 3, AC394

STA 0, RM94

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

LCLS4: JPS GNC67

LEA 1, CHZ94

LEA 2, CHF94

ADCZ 2, 0, 0

ADCZ 0, 0, 0

JMP CHG94

JMP MA994

IF NOT NUMERIC, CHECK IF IT IS IN A-F.
NAME: \text{OCT42}

FUNCTION: Converts a series of ASCII DEcimal CH. into a binary NUMBER.

INPUT: CHPTA points to CH. before SING. CH. are requested thro A CALL to GNC7.

OUTPUT: CHPTA points at \text{BREAK}.

CALLS: GNC7.
CALLER.JV.
GLOBAL-VARIABLES-USED. NONE.
GLOBAL-VARIABLES-CHANGED. NONE.
ERROR-BITS-SET. NONE.

SPACE FOR SAVING ACC CONTENTS.
AC21: BLK 1
AC31: BLK 1

CONSTANTS
CM32: 67 ; CH 7
C12: 60 ; CH 9
FSM92: BLK 1

SAVE ACC ENTRY, INITIALISE RUNNING SUM.
JR1921 SIA 3,AC292
STA 3,AC392
SUB 3,FSM92

GET NEXT CH AND UPDATE RUNNING SUM.
LEL921 JS26 CHC97
LDA 1,CHC92
LDA 2,CHC92
ADC 2,9,CNC
AC993 0,1,CZC
JMP BK92

SUB 1,0
LDA 1,FSM92

MULTIPLY BY 6 AND ADD
MVI ZL 1,1
MOVZL 1,1
MOWZL 1,1
ADS 0,1
NAME: DEC21.

FUNCTION: CONVERTS A SERIES OF ASCII OCTAL CH. INTO A BINARY NUMBER.

INPUT: Optr points to CH.

Before STORE, Ch. are reflected thro. a CALL to O2837.

OUTPUT: Optr points at break.

ACC 0 contains break.

ACC 1 contains value.

CALLS: O2837.

CALLED BY: CVT34, PAG5, MPR17.

GLOBAL-VARIABLES-USED: none.

GLOBAL-VARIABLES-CHANGED: none.

ERROR-BITS-SET: none.

SPACE FOR SAVING ACC CONTENTS:

ACC 0 contains BLK 1

ACC 1 contains BLK 1

CONSTANTS:
SAVE ACC CONTENTS, ENTRY POINT, INITIALISE RS 31.

DE911: STA 3, AL911
STA 3, CH911
SUB 9, 9
STA 9, RS911

EVALUATE BY SUCCESSIVE CALLS TO CMQ7 AND MULTIPLYING BY 10.

LE911: JS< CMQ7
LOA 1, CM911
LDA 9, CH911
ADC 0, 6
JMP 9, RS911

CH IS NOT BREAK

SUB 1, 9
LOA 1, RS911

MULTIPLY BY 10 AND ADD

MOVZL 1, 2
MOVZL 2, 2
ADC 9, 1
MOVZL 1, 1
ADC 0, 1

STA 1, RS911
JMP LEL91

ACC 0 CONTAINS BREAK CH, RETURN

BE911: LOA 1, RS911
LOA 2, 4C91
JMP AC911
NAME.
SYMB.

FUNCTION.
GETS THE VALUE OF
THE SYMBOL FROM SYMB.

INPUT.
CHAR POINTS TO CH.
BEFORE SYMBOL.

OUTPUT.
CHAR POINTS TO BREAK CHAR.
ADD 0 CONTAINS BREAK (=0 IF
CHAR).
ADD 1 CONTAINS VALUE.

CALLS.
STAM, SHORT, ENDB.

CALLS-BY.
CTJ 36.

GLOBAL-VARIABLES-USED.
GLOBAL-VARIABLES-CHANGES.
LOCK, PSLG.

ERROR-BITS-SET.
6(5), 6(0).

SPACE FOR SAVING ADD CONTENTS.
AC2951 : BLK 1
AC3951 : BLK 1

CONSTANTS
A2751 52
C2751 45
C15951 8
CT551 7

COUNTERS AND SPACE FOR STORING LABEL
CT551 : BLK 1
LML351 : BLK 3
LB4951 : LBL95

SAVE ADD CONTENTS, ENTRY
SET UP COUNTER TO COUNT UNTIL 6

LDA 0, CJT 95
STA 0, CTR 95

CLEAR LABEL SPACE

LDA 0, 3295
SUB 0, 0
STA 0, 1
STA 0, 12

GET CH AND SAVE

LDS 111, CHSE
JSR CS 97
JMP BRK 95

CH IS ALPHANUMERIC, CHECK FOR LENGTH

DSZ CTR 95
JMP +2
JMP ERR 95

CHECK HIGH HALF

LDA 3, 3CT 95
LOA 0, 11
JMP ZLY 95

CARRY IS NON ZERO, STORE AND UPDATE ACC 2

ADD 1, 1
STA 0, 1, 2
INC 2, 1
JMP LDP 95
CARRY IS ZER0, SAVE IN ACC 1

ZCY55: MOVs 0,1
JMP LCP95

CH IN ACC 0 IS BREAK, SEARCH TABLE AFTER POSSIBLY SAVING ACC 1

BLK51: LOA 3, CT55
STA 3, 245
LOA 3, 532, H
STA 1, J12

LOA 1, L5435
STA 1, G6D99
STA 1, 150195
JSR 63137I

AGC55: *BLK 1
AGC55: *BLK 1
AGC55: *BLK 1

CHECK WHETHER FOUND OR NOT

LOA 3, AG235
LOA 3, FL595
MOVLE 2, 2532HC
JMP END55

NOT FOUND, ENTER ERROR NO 5
IF PASS IS 2, SET NULL BREAK CHARACTER.

LOA 2, B5SEL
LOA 2, B5S195
JMP SET595

IF IT SET ERROR BIT.

PASS IS 2, SET ERROR BIT 5.

LOA 2, CM635
STA 2, AG695
JSFS 24225
AG395:  J.1K  1

SET36:  SUB  0,0
        SUB  1,1
        JMP  EN395

SYMBOL TOO LONG, ENTER ERROR BIT 6.

ERR46:  LDA  2,CH395
        STA  2,AC395
        JSR  EN226

AG466:  J.1K  1
        SUB  1,1
        SUB  0,0
        JMP  EN395

ERR51:  LDA  1,CH395
        JSR  EN327

RESTORE ACC, RETURN.

ERR51:  LDA  2,AC395
        JMS  2,AC395

NAME:   EVAL37
FUNCTION:  EVALUATE AN EXP.
INPUT:   CHA PRT: POINTS TO FIRST CH. OF EXPN.
OUTPUT:  CHA PRT: POINTS TO BREAK.
          ACC 0 CONTAINS BREAK, NULL
          IF ERROR.
          ACC 1 CONTAINS VALUE.
CALLS:  EN226, CS395, AC395;
SUB39, WDIV395, DIV395.
CALLED BY:  L57.
GLOBAL-VARIABLES-USED:  N/A.
GLOBAL-VARIABLES-CHANGED:  N/A.
SPACE FOR SAVING ACC CONTENT.

ACC97:  .BLK  1
AM97:   .BLK  1

SPACE FOR BREAK, VALUE
AND ADDRESS OF SERVICE ROUTINE.

AMK97:  .BLK  1
B97:     .BLK  1
V97:     .BLK  1

SAVE ACC CONTENT, ENTRY, SET CMPTR.

EV197:  STA  2, AC297
        STA  3, AC397
        JSR  CMPTR

INITIALISE VAL97, 3R97 TO 0, +

STA  0, J
LOA  0, VAL97
STI  0, 3R97

CHECK 3BREAK CHARACTER.

LCL37:  LOA  0, 3=K97
        MVE  0, J, SZE
        STA  0, 3=BREAK

CHECK IF I IS +

LOA  1, PLS97
LDA 1,1567
SUBE U1,SNF
JMP SN37

C H E C K  I F  I T  I S  0.
LDA 1,15697
SUBE U1,SNF
JMP SN97

C H E C K  I F  C H.  I S  +.
LDA 1,45T97
SUBE U1,SNF
JMP NL97

C H E C K  I F  I T  I S  B L A N K  O R  C O M A.
LDA 1,3L797
SUBE U1,SNF
JMP NL97
LDA 1,00097
SUBE U1,SNF
JMP NL97

C H A R A C T E R  I S  N O N E  O F  A B O V E,  S E E  I F  I T  I S
A  N U L L,  I F  S O  D O  N O T  S E T  E R R O R  B I T  S I N C E
I T  H A S  S E E N  A L R E A D Y  S E T.

MOV 0,J,SNF
JPF ER97

E X P R E S S I O N  E R R O R.

LDA 0,SEV97
STA ER97
JMP ER97

ACC97; J,0LK 1

S E T  A C  0  A N D  A C  1  T O  5.
E-197: SUB: 0,1
SUB 1,1
JIP EN397

ADDITION SYMBOL. SET ADDRESS.
ADD74: LDA 1,40397
STA 1,40397
JIP GVL97

SUBTRACTION SYMBOL. SET ADDRESS.
SUB74: LDA 1,30399
STA 1,40397
JIP GVL97

MULTIPLICATION SYMBOL. SET ADDRESS.
MUL74: LDA 1,4P778
STA 1,40297
JIP GVL97

DIVISION SYMBOL. SET ADDRESS.
DIV74: LDA 1,2IL93
STA 1,40297

CALL ON ROUTINE TO CONVERT NEXT OPERAND.
GVL97: JSRS GVT96

ACC 0 CONTAINS BREAK CHARACTER.
ACC 1 FOLLOWS VALUE.
ACC 3 CONTAINS NULL IF EXP CANNOT BE EVALUATED.

STA 0,3FK97

PERFORM OPERATION:
ACC 3 CONTAINS FINAL RESULT.
ACC 0 AND ACC 1 ARE INPUTS USED BY ROUTINE.

LOAD 0,4VAL97
JPS ACC97
STA 0,4VAL97
JIP LBL97

CH. IS BLANK, END OF EXP.
BREAK ON CH IS IN ACC 0, RETURN.

LXI 37: LDA 1,VAL 37
EIX 37: LDA 2,AC37
JMPS AL37

NAME.
FUNCTION.
Evaluates the next operand if possible.

INPUT.
Pointer pointing to CH before symbol number.

OUTPUT.
Pointer points to break.
Value in ACC 1.
Check in ACC 0, null if error.

CALLS.
AC37, PXX, I, D, E, C;
AC37, PXX, G, D, E;

CALLED BY.
AC37.

GLOBAL-VARIABLES-USED.
NONE.

GLOBAL-VARIABLES-CHANGED.
NONE.

ERROR-BITS-SET.
NONE.

SPACE FOR SAVING ACC CONTENTS.
AC2561 .BLK 1
AC2961 .BLK 1

CONSTANTS.

TEXT:
AC2561 45
ATT361 15
DCL361 44
BLK561 40
CH7361 60

; :.
% ; 
; 
; ;
; ;
CH.38: 71
CH.39: 54

SAVE ACC CONTENTS, ENTRY POINT

CH.39: STA 2,40296
STA 3,40396

GET FIRST CHARACTER OF NUMBER, SYMBOL AND DECIDE WHICH ROUTINE TO CALL.

LSA 0,0537
LSA 1,PEN3
SUL5 9,11,1SHK
JMP BIN96

; CH IS %

LSA 1,3053
SUL5 9,11,3SHK
JMP HEX96

; CH IS 8

LSA 1,41196
SUL5 9,11,4SHK
JMP OCT96

; CH IS <

CHECK IF CH IS BLANK OR COMMA

LSA 1,8BLK36
SUL5 9,11,5SHK
JMP NXT96

; CH IS BLANK

LSA 1,6COM36
SUL5 9,11,6SHK
JMP NXT96

; CH IS COMMA

CHECK IF DECIMAL OR SYMBOL

LSA 1,2HZ96
LSA 2,C4LB36
ASCZE 2,1JSHK
ASCZE 9,11,SZK
JMP SYM96

; IT IS A SYMBOL

CH. IS BETWEEN 0-9, TRANSFER TO CONVERSION ROUTINE

AFTER ADJUSTING CHPTW.

DSP 1,CHPTW
JMP 1,DECO1
JMP EN39A

CH IS BINARY,-transfer to routine.

DINE: JOS 3RY 93
       JMP EN396

CH IS HEXADECIMAL.

HEX93: JOS 3HXY 94
       JMP EN396

CH IS OCTAL.

OCT94: JOS 3OCTY 92
       JMP EN396

CH IS SYMBOL.

SY:95: JOS 3CHYP
       JMP SYM 95
       JMP EN396

CH IS BLANK or COMMA, set value to 0.

NX396: SUB 1,1
      RESD3 4 ACC CONTENTS, return.

END96: LRA 2,4096
       JMP AC396

NAME.

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: EVL.
GLOBAL VARIABLES CHANGED: NONE.
ERROR BITS SET: NONE.

SPACE FOR SAVING ACC CONTENTS.

ACC72: .BLK 1
AC372: .BLK 1

CONSTANTS:

CHS72: -21

ENTRY POINT, SAVE ACC CONTENTS.

MP172:
SIA 3,AC372
SIA 2,AC272

SUBS 2,2
LDA 3,0H372

: 16 TIMES THRO THE LOOP.

LABEL72:
MOVH 0,0,SHC
MOVH 2,2,SKP
ADDZ 1,2
INC 3,3,SZK
MOV LBL72
MOVCH 0,0

: YES, SHIFT LAST BIT.

RETURN.

LDA 2,AC272
JMP AC372
NAME.

FUNCTION.

INPUT.

OUTPUT.

CALLS.

CALLED-BY.

GLOBAL-VARIABLES-USED.

GLOBAL-VARIABLES-CHANGED.

ERROR-BITS-SET.

SPACE FOR SAVING ACCS.

AC1991  .BLK  1
AC2991  .BLK  1
AC3991  .BLK  1

CONSTANT

CHEX91  2
MUL991  43
LIT991  47
MX991  377

SAVE ACC CONTENTS.

IM1991  STA  1,AC135
         STA  0,AC135
         STA  3,AC135
         STA  2,AC135

; CHECK IF CH IS F
CH IS E. CHECK IF NEXT IS 
  J 7, 0067
  L O A 1, 3 L I T 99
  S U R E J, 1, 1 S Z R
  J M P N X 99

CH IS A LITERAL, STORE IN WSPCE AS IT IS.
  J S R 7, 0067
  S T A 1, 3 W S P C E
  M C V S 0, 1
  J S R 7, 0067
  A D D 0, 1
  L O A 1, 3 W S P C E
  S T A 1, 1, 5
  S U R 0, 0
  L O A 1, 1, C H 239
  S T A 1, 1, E N T H
  J M P E N 599

CH IS NOT LITERAL, EVAL EXPRESSION.
  E X P 99 J S R 7, 0067
  L O A 0, 3 W S P C E
  S T A 1, 3 W S P C E
  L O A 1, 3 W S P C E
  S T A 1, 1, 5
  S U R 0, 0
  L O A 1, 1, C H 239
  S T A 1, 1, L E N T H
  J M P E N 599

SET TO 1 ACC J
  N E X T 99 S U R 0, 0
  I N C 0, 0

RESTORE ACC CONTENTS.
  E N G 99 L O A 1, A C 1 9 9
NAME: BCC70.

FUNCTION: SERVICES THE RELATIVE MODE OF ADDRESSING, EVALUATION.
INSTRUCTIONS SERVICED ARE:

BCS BCS
d BE BE
BLE BLE BLE
BE BE

INPUT: CHAIN POINTS TO O.M. AFTER iP-llie.

OUTPUT: LENGTH, HEBCE GET.
CHAIN POINTS TO BREAK.

CALLS: EVL67, EV62.

CALLED-BY: SP-29.

GLOBAL-VARIABLES-USED: MSPACE, LOCTR, ISVA, AUF6G.

GLOBAL-VARIABLES-CHANGED: LENGTH, MSPACE.

ERROR-BITS-SET: 15(F).

SPACE FOR SAVING ACC CONTENTS:

AC701: BLK 1
AC701: BLK 1
AC701: BLK 1
AC701: BLK 1

CONSTANTS:

CA701: 2
CA701: 4
CA701: 12
MSK701: 377
; SAVE ACC CONTENTS.

; LOC:       STA  J,ACU70
           STA  5,ACU70
           STA  3,ACU70

; IF ABFLG IS SET, PROCEED
; IF ABFLG = 3, PROCEED; OTHERWISE FOR ANY ERROR.

JSR5  SCP23
JMP  END7
LDA  L,ABFLG
LDA  1,CH770
SUB3  2,1,CZR
JMP  ERR7

; EVALUATE EXP

JSR5  EVL97

; SET LENGTH

STA  2,CH770
STA  2,LENGTH

; GET VALUE

STA  2,LOUTF
LOAD  2,1,6370
LOAD  2,1,MSK70
AND  2,1,SAL
LDA  1,CH770
STA  2,MSCE
JMP  END7

; LOC:       STA  J,ACU70
           STA  5,ACU70
           STA  3,ACU70

; RESTORE ACC, RETURN

END7  LDA  J,ACU70
NAME: EXPZ
FUNCTION: SERVICES EXTENDED CODE OF ADDRESSING.
INPUT: CIPTR POINTS TO FIRST CH. OF EXPRESSION.
OUTPUT: CIPTR POINTS TO SPARE.
LENGTH SET TO 0.
NSPACE SETS 2 AND 3 SET.
CALLS: EVLPZ.
CALLED-BY: LUTS.
GLOBAL-VARIABLES-USED: NSPACE.
GLOBAL-VARIABLES-CHANGED: NSPACE, LENGTH.
ERROR-BITS-SET: NONE.

SPACE FOR SAVING ACC CONTENTS.
ACC 021: 1 BLK 1
ACC 022: 1 BLK 1
ACC 023: 1 BLK 1
ACC 024: 1 BLK 1

CONSTANTS:
C 021: 3
MKP 021: 377

SAVE ACC CONTENTS.
E1 021: STA 4,40312
STA 1,A1002
STA  2,42:02
EVALUATE EXP.

SET LENGTH

LOA  2,33:02
STA  2,LENGTH

SET MSPACE

LOA  3,MSPACE
LOD   3,1K-02
STA   3,1,2
MOVS  1,1
STA   1,3

RESYDE AND CONTENTS.

LOA  2,40:02
LOA  1,A1:02
LOA  2,A2:02
JMP$  A1:02

NAME.
FUNCTION.
SERVICES THE DIRECT MODE
OF ADDRESSING.

INPUT.
DMPTR POINTS TO FIRST CH.
OF EXPRESSION.

OUTPUT.
DMPTR POINTS TO BREAK.
LENGTH SET, MSPACE SET.

CALLS.

CALLED-BY.
EVLST.
GLOBAL-VARIABLES-USED:  ARGC, DGRA, PSEFL, GPELT, LCTR.
GLOBAL-VARIABLES-CHANGED:  DGRA, PSEFL, LENGTH.
ERROR-BITS-SET:  NONE.

SPACE FOR SAVING ACC CONTENTS:

<table>
<thead>
<tr>
<th>ACC</th>
<th>BLK</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC001</td>
<td>1</td>
</tr>
<tr>
<td>ACC002</td>
<td>1</td>
</tr>
<tr>
<td>ACC003</td>
<td>1</td>
</tr>
<tr>
<td>ACC004</td>
<td>1</td>
</tr>
</tbody>
</table>

CONSTANTS:

| C0001 | 377  |
| C0002 | 33   |
| C0003 | 62   |
| C0004 | 177400 |

SAVE ACC CONTENTS:

| B0001 | STA 1, A0000 |
|       | STA 1, A1000 |
|       | STA 1, A2000 |
|       | STA 1, A3000 |

EVALUATE EXPRESSION:

| 0000 | JMR  EVL97  |
|      | STA 1, J8000 |

AC 1 contains value, AC 1 contains break.
AC 0 = j if error in evaluation.

| 0000 | MDJ  3, J, SR |
|      | JHF  EX10   |
|      | LOA  3, PSEFL |
|      | LOA  3, 02, 00 |
|      | SUBE  1, 2, CR |

194
JMP PASS1

PORT = 1, CHECK IF DIRAD IS FULL, AC 1 CONTAINS VALUE
LDA 1, DIRAC
LDA 2, C0100
AND 1,2
LDA 3, DREPT
SUBE 2, DREPT
JMP EX500

PORT = 2, CHECK IF AC 1 BETWEEN J0 AND FF.
LDA 0, CSJ0
AND 1,2
JMP EX500

PORT = 1, DIRAD IS NOT FULL, VALUE LESS THAN FF.
LDA 0, L1CT
STA 0, DREPT
LDA 3, DREPT
JMP DR500

PORT = 2, CHECK IF LINE PRESENT IN DIRAD, IF SO DIRECT ELSE EXTENDED.
PORT = 1, NOT FOUND
SUBE 2,3, SHF
JMP EX500
LDA 1, J1,2
SUBE 1,1, SHF
JMP DR500

PORT = 2, FOUND
JMP EB300

ENTER WORD, SET LENGTH, STORE VALUE.
EXEC: LDA 0, C0100
STA 0, CS100
LDA 2, SPACE
LDA 1, V1000
LDA 0, C1000
DIRECT MODE, SET LENGTH, STORE VALUE

ENDU1  LOA  1,VL3U3
      STA  1,C13UJ
      ADD  1,J
      STA<  0,NSPCE
      STA  2,DEBUJ
      I$Z  A33U5

STORE ACC AND RETURN.

ENDU1  LOA  0,AL3UJ
      STA  1,A33U5
      LDA  1,A33U5
      JMP$  A33U0

NAME.

FUNCTION.

SERVICES THE INDEXED MODE OF ADDRESSING.

INPUT.

CHARACTER POINTS TO FIRST CHARACTER OF EXPR.

OUTPUT.

CHARACTER POINTS TO BREAK.

IF INDEXED, CHAR POINTS TO BREAK, ACC J = 0, LENGTH SET TO 2, NSPCE SET.

IF NOT INDEXED, ACC J = 1, CHAR UNCHANGED.

CALLS.

SCHU1, DUC7, EVL97, SDC41.

CALLED BY.

LUC.
GLOBAL-VARIABLES-USED:

GLOBAL-VARIABLES-CHANGED:

CHECK-HITS-SET:

SPACE FOR SAVING ACC:

A1: BLK 1
A2: BLK 1
A3: BLK 1
TME: BLK 1

CONSTANTS:

C1: 121
C2: 2
C3: 2614
C4: 5494
M: 577

SAVE ACC CONTENTS, ENTRY

I1: ST: 1,41301
     BA: 8,12601
     TA: 3,45501

CHECK IF ERROR

LOAD 0,CHT=1
LOAD 1,ST=01
SMB 0,1,SN=1
JMP NIS=1

CHECK IF CH=1

JMP 0
MO: 0,6
JMP 1,CH=0
ACC 0,2
LOAD 1,ST=01
JMP 0,SN=1
JMP L=501
GET NEXT NON-BLANK CHARACTER.

LDY CHPT
STA JS
LDA JS
STA JS1
LDA CLO
STA CHPT
LDA CHPT
STA GCHE
LDA JS
STA JS1

CHECK IF $, $.

LDA JS3
SUB JS
JMP NISU1

INDEXED,
EVALUATE EXP AND SET LENGTH, AC 0, NISU1

LDA JS
STA CHPT
STA JS
LDA JS
STA JS
LDA JS
STA JS
LDA JS
STA JS
AND JS
SUB JS
JMP NISU1

NOT INDIRECT, SET AC 0

NISU1
SUB JS
INC JS
LDA JS3
STA JS
LDA JS
STA CHPT

RESTORE AC0 AND RETURN.

End11
LDA JS3
LDA JS3
JMP A33J1
NAME:  SUB328

FUNCTION: SETS ABFLG IF IT IS NOT ALREADY SET.
          SET IT TO EITHER 2 OR 1.

INPUT:  CHTR POINTS TO CH. AFTER OP-CODE.

OUTPUT: ABFLG SET TO 2 OR 3;
        CHTR POINTS TO FIRST CH.
        OF EXPRESSION;
        RETURNS TO ADD 1 IF ERROR;
        RETURNS TO ADD 3 + 1 OTHERWISE.

CALLS:  SCB41, SCB32.

CALLED-BY: LOTE.

GLOBAL-VARIABLES-USED: none.

GLOBAL-VARIABLES-CHANGED: ABFLG.

ERROR-STATE-SET: 1, (F),

SPACE FOR SAVING ADD CONTENTS.

ACCESS:  .BLK 1
          .BLX 1
          .BLK 1
          .BLK 1

CONSTANTS:

  CR3981  .12 ; DEC 10
  CR2581  .3
  CR1981  .1
  CR1581  .121 ; DEC 1
  CR1581  .40440 ; DEC 1
ENTRY POINT, SAVE ACC CONTENTS.

SEE IF ADFLG IS 2 OR 1

IF ADFLG IS NOT SET, CALL SCP23 TO SET CHPTF TO NEXT CHAR BLANK CHARACTER.

; ERROR.

; ERROR(IF)-HALT, BIT NO. 1 IN BIT.
CH IS 2A

OKA3:
LDA 1,CH13
STA 1,HSD
ISZ 1,AC193
JMP END93
JMP END93

CH IS 2B

OKA3:
LDA 1,CH13
STA 1,HSD
ISZ 1,AC193
JMP END93
JMP END93

SET CMPTR

SET93:
JSA 1,LSZ
JSA 1,SCP23
JMP END93
ISZ AC193

RESTORE ACC AND RETURN.

END93:
LDA 1,AC193
LDA 1,AC193
LDA 1,AC193
JMP AC193

---------------------------------------------------------------
NAME.
FUNCTION.
INPUT.
OUTPUT.
CALLS.

ACC 0 = ACC 0 + ACC 1.
ACC 0,ACC 1.
ACC 0.
NONE.
GLOBAL-VARIABLES-USED: EVAL77.
GLOBAL-VARIABLES-CHANGED: NONE.
ERROR-BITS-SET: NONE.

01161 ADD 1,1
      JMP 0,9

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

NAME: SUB33.
FUNCTION: SUB33.
INPUT:  ACC 0 = ACC 0 + ACC 1.
OUTPUT: ACC 0, ACC 1.
CALLS:  ACC 0.
CALLED-BY: EVAL77.
GLOBAL-VARIABLES-USED: NONE.
GLOBAL-VARIABLES-CHANGED: NONE.
ERROR-BITS-SET: NONE.

SU189: SUB: 1,1
      JMP 0,9

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

NAME: DIV30.
FUNCTION: DIV30.
INPUT:  ACC 0 = ACC 0 / ACC 1.
OUTPUT: ACC 0, ACC 1.
OUTPUT:

CALLS:

CALLED BY:

GLOBAL VARIABLES USED:

GLOBAL VARIABLES CHANGED:

ERROR BITS SET:

SPACE FOR SAVING ACC CONTENTS:

AC250:

AC350:

CONSTANTS

CHECK:

ENTRY POINT, SAVE ACC

CI190:

CHECK ACC 1 IF IT IS J

MCY 1, SH

MP ERR 90

ACC 1 IS NOW ZERAO, CLEAR ACC 2, SET UP COUNTER FOR 16

SUB 2, 2

LOA 3, .CHS93

MOV L 0, V

!SHIFT LOW DIVIDEND.

LNL90:

MCYL 2, 2

!SHIFT HIGH DIVIDEND.

SUB 1, 2, .SZC

!DOES DIVIDEND 60 IN.
ERROR BITS SET.

SPACE FOR SAVING ACC CONTENTS.

LOC 631 $BLK 1
LOC 631 $BLK 1
LOC 631 $BLK 1
LOC 631 $BLK 1

CONSTANTS.

FOR 631 4
THO 631 2

SAVE ACC ENTRY

ADD 631 STA 0,4000
STA 1,4016
STA 2,4020
STA 3,4036

CALL ON ROUTINE TO SET SHFT, ASFLG.

JMP CALL
JMP END

CHECK WHICH OF THE FOUR ADD EQUATING TYPES.

SUB 1,1
JMP IM 1
MIV 0,1,SHF
JMP IM 1

JMP TNG 1
MIV 0,1,SHF
JMP TNG

JMP DTTJ
JMP EXT 1
JMP DIT 1

EXT 1 INC 1,1
TNG 1 INC 1,1
015621  INC  1,1

: CHECK WHETHER A OR B.

015631  LOA  0,AC66
015631  LOA  0,AC66
015631  LOA  3,AC66
SUB  3,AC66
ADO  2,1

: FORM FIRST BYTE.

MOVZ  1,1
MOVLZ  1,1
MOVLZ  1,1
MOVZ  0,USVAL
MOVZ  0,MSPACE
ADO  1,0
STAS  0,MSPACE

: RETURN.

015631  LOA  1,AC66
015631  LOA  1,AC66
015631  LOA  3,AC66
JMPS  AC66

------------------------------------------------------------------
NAME:  R1361
FUNCTION: SERVICES THE PSEUDO-UP, RMB.
ALLOCATES STORAGE SPACE.
INPUT:  NONE.
OUTPUT:  NONE.
CALLS:  STR04, EVL07, EHE20.
CALLED-BY:  SP-25.
GLOBAL-VARIABLES-USED:  DCE.
GLOBAL VARIABLES CHANGED: VALUE, IFLG, PFLAG.
ERR04 - RTS - SET.
11(8).

SPACE FOR AVOIDING ACC CONTENTS

ON ENTRY CHPT E IS SET TO CHAFTER OPCODE

AC1611 .BLK 1
AC1612 .BLK 1
AC2611 .BLK 1
AC3611 .BLK 1

CONSTANTS

EVAR 11: 13

SAVE ACC, ENTRY

PM1611 STA 3,AC3611
          STA 1,AC1611
          STA 2,AC2611
          STA 3,AC3611
          ISZ VFLG

SET CHPT E AND EVALUATE

JSR= ST3994
JMP= EN061
JSR= EVL37

CHECK IF IT CAN BE EVALUATED

MCL 4,VALUE
JMP= ERR91

SET LOCTR AND PFLAG

STA 1,VALUE
LDA 0,LOCTR
ADD 1,1
STA 0,LOCTR

ISZ PFLAG
JMP= EN 91
SET BIT 11

ERR11: LDA 9; EV, 61
STA 3, EV, 61
JNE< PA, 29
LDA 1; BLK 1

RESTORE ACC AND RETURN

ERR11: LDA 1, ACC, 61
LDA 1, AC, 161
LDA 2, AC, 261
JMPS AC, 361

NAME: 0-656
FUNCTION: This routine services the Pseudo-or flag, sets the
program counter.
INPUT: None.
OUTPUT: None.
CALLS: None.
CALLED-BY: STCA, EV, 07, ERR26.
GLOBAL-VARIABLES-CHANGED: LDFG.
ERROR-BITS-SET: 12(H), 11(H).
SPACE FOR SAVING ACC CONTENTS:

AC356: .BLK 1
AC156: .BLK 1
AC256: .BLK 1
AC056: .BLK 1
; CONSTANTS
TWL59: 14
ONE59: 13
EVN59: 1

; SAVE ACBS. ENTRY
OEE159:
STA 0, AC159
STA 1, AC159
STA 2, AC159
STA 3, AC159

; CHECK IF LABEL IS PRESENT
LDA 0, LSFALG
SUB 0, ONE59
JMP ER259

; SET CHPTK AND EVALUATE EXP.
JFS S, ST6E
JMP EN655
JSRS EVL67
MCV 0, SNF
JMP ERR59

; STA 1, LOCTA
STA 1, LOCTN
LND FNFLG
JMP EN259

; ER69, SET BIT NO. 12
ER259: LDA 0, TWL59
STA 0, T6559
JGO ER265
AG259: JBLK 1
JMP END59

; SET ERROR BIT NO. 11(R).
ER59: LDA 0, EW59
STA 0, E659
JGO ER269
AG59: JBLK 1
NAME:  
FUNCTION:  
INPUT:  
OUTPUT:  
CALLS:  
CALLED-BY:  
GLOBAL-VARIABLES-USED:  
GLOBAL-VARIABLES-CHANGE:  
ERROR-BITS-SET:  

SPACE FOR AVING ACS CONTENTS:

AC531  BLK  1
AC1531 BLK  1
AC2531 BLK  1
AC3531 BLK  1

CONSTANTS

ONE531  1
ONE531  1

SAVE ACS ENTRY

EC1531 STA 1,AC1531
STA 1,AC1531
STA 2,AC3531
STA 3,AC3531
LSZ 1LFLG
SUB 1;1
STA 1;LCFLG

CHECK IF THERE IS LABEL IF SO OKAY

LDA 3;LBFILG
JCV J0,;SHR
JMP ERR53

SET CMPTR AND EVALUATE

JSRS ST24
JMP EN353
JSRS EVL97

CHECK IF EVALUATED PROPERLY

JCV J0,;SHR
JMP IPP53

EVALUATION IS OKAY SET VALUE

LDA 3;ABDLBL
STA 1;3;3
STA 1;VALUE
JMP EN353

SET BIT NO. 9

ERR53: LDA 3;IN353
STA 3;INF53
JERK ERR26

INF53: JLRK 1
JMP EN353

EXP. CANNOT BE EVALUATED SET FIRST NO TO ZERO

INF53: LDA 3;ABDLBL
SUB 0;3;
STA 0;3;

RESTORE Acc AND RETURN

END53: LDA J;ACJ53
LDA 1;AC153
LDA 2;AC253
JMPS   AC314

FUNCTION  SEP14.
INPUT.   OR THE FOLLOWING VARS.
         PIERR, PRERR.
OUTPUT.  NONE.
CALLS.  NONE.
DIRECTORY.  DEP77.
GLOBAL-VARIABLES-USED.  PAGE2.
GLOBAL-VARIABLES-CHANGED.  PIERR.
ERROR-BITS-SET.  NONE.

SPACE FOR ACC CONTENTS
AC314.  BLK  1

ENTRY POINT
CE314:  STA  3,AC314

CE314:  LOA  3,PRIERR
CE314:  STA  3,AC314
CE314:  LOA  3,PRERR
CE314:  STA  3,AC314
CE314:  JSLX  OR77

AC314:  BLK  1
AC314:  BLK  1
AC314:  BLK  1
AC314:  LOA  3,AC314
AC314:  STA  3,PRERR

RETURN
NAME: AC314
FUNCTION: SERVICES THE INSTR. WITH FORMAT NO. 1.
INPUT, OUTPUT: CHART POINTS TO BREAK, WPCE, LENTH SET.
CALLS: SADD, SBIND, INSL.
CALLED-BY: SRM25.
GLOBAL-VARIABLES-USED: BAVAL, ACPLG, RCP01.
GLOBAL-VARIABLES-CHANGED: RCP01.
ERROR-BITS-SET: NO.

SPACE FOR SAVING ACC CONTENTS.
AC064: BLK 1
AC164: BLK 1
AC264: BLK 1
AC364: BLK 1

CONSTANTS:
IN064: 2
FCF64: 4

SAVE ACC CONTENTS, ENTRY:
ST164: STA 0, AC064
ST164: STA 1, AC164
ST164: STA 2, AC264
ST164: STA 3, AC364

CALL ON ROUTINE TO SET CHART AND ABELS.
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSKS</td>
<td>CASB364</td>
</tr>
<tr>
<td>JMP</td>
<td>EHS64</td>
</tr>
</tbody>
</table>

CHECK WHICH MODE AND SET ACC 1 ACCORDINGLY:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JSKS</td>
<td>1,7,1</td>
</tr>
<tr>
<td>JSKS</td>
<td>1,10,1</td>
</tr>
<tr>
<td>MOV</td>
<td>0,1,SNK</td>
</tr>
<tr>
<td>JMP</td>
<td>1,16364</td>
</tr>
<tr>
<td>JMP</td>
<td>DIS64</td>
</tr>
<tr>
<td>JMP</td>
<td>EX64</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXT64:</td>
<td>INC 1,1</td>
</tr>
<tr>
<td>INC64:</td>
<td>INC 1,1</td>
</tr>
</tbody>
</table>

CHECK WHETHER ACC A OR ACC B:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>0,13FLG</td>
</tr>
<tr>
<td>LDA</td>
<td>2,10564</td>
</tr>
<tr>
<td>LDA</td>
<td>3,10564</td>
</tr>
<tr>
<td>ADD</td>
<td>3,1,SNK</td>
</tr>
</tbody>
</table>

FORM FIRST BYTE:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOV</td>
<td>1,1</td>
</tr>
<tr>
<td>MOV</td>
<td>1,1</td>
</tr>
<tr>
<td>MOV</td>
<td>1,1</td>
</tr>
<tr>
<td>MOV</td>
<td>3,1</td>
</tr>
<tr>
<td>LDA</td>
<td>0,1SVAL</td>
</tr>
<tr>
<td>ADD</td>
<td>0,1</td>
</tr>
<tr>
<td>ADD</td>
<td>0,1</td>
</tr>
<tr>
<td>STA</td>
<td>0,1</td>
</tr>
<tr>
<td>STA</td>
<td>0,1</td>
</tr>
</tbody>
</table>

RESTORE ACC AND RETURN:

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>0,AC364</td>
</tr>
<tr>
<td>LDA</td>
<td>1,AC164</td>
</tr>
<tr>
<td>LDA</td>
<td>2,AC364</td>
</tr>
<tr>
<td>JMP</td>
<td>AC364</td>
</tr>
</tbody>
</table>

NAME:
FUNCTION.

ASL15.

SERVICES THE INST. WITH
FORMAT NO. 2.
ASL ASL C33 C01 DEC
INC LEH NEG ROL ROR TST.

INPUT.

COPY POINTS TO CH. AFTER
OP-CODE.

OUTPUT.

COPY POINTS TO BREAK,
NSPCE,LENGTH SET.

CALLS.

$8223, DEC, DEC, DEC,
INC, EX, EX.

CALLED-BY.

$6425.

GLOBAL-VARIABLES-USED.

GLOBAL-VARIABLES-CHANGED.

ERROR-BITS-SET.

NONE.

SPACE FOR SAVING ACC CONTENTS.

ACC15: 6 BLK 1
AC15: 6 BLK 1
AC35: 6 BLK 1
AC35: 6 BLK 1

CONSTANTS.

$655: 44444 41040 3
ONE: 555 1
TWO: 555 2

SAVE ACC CONTENTS, ENTRY.

$615: STA 0, AC155 STA 1, AC155
STA 2, AC355 STA 3, AC355
STA 0, ONE55 STA 0, LENGTH

215
CHECK IF ABFLG IS SET:

SUB 1,1
LDA 2,ABFLG
LDA 2,UNFLG
SUI 0,3,SR
JMP AS165
LDA 2,4,SR
SUI 0,3,SR
JMP AS165

ABFLG IS 1, SEE IF A OR 3 IS PRESENT.

JMP< SOR23
JMP< ESO65
JMP< GCH31
MOV< 0,KR
JMP< GH387
ADD< 2,6
LXI< 2,CHA6B
SUI< 0,3,SR
JMP AS165
LDA< 2,GCH345
SUI< 0,3,SR
JMP AS165
JSZ CORTK

IT IS NOT A,3, SEE WHAT TYPE OF ADDRESSING.

JMP< Ih361
MOV< 4,SR
JMP< IH64
JMP< EX362
INC< 1,1
INC< 1
INC< 1,1

FORM FIRST BYTE

LXI< 1,1
MOV< 1,1
MOV< 1,1
MOV< 1,1
LDA< 0,BOAL
ACCS  U,1
ACCS  U,SPACE
ACCS  1,SPACE

RETURN AFTER SAVING ACC CONTENTS.

ENDS:  LOAD  0,ABS
LOAD  1,ABS
LOAD  2,ABS
JMPS  ABS

NAME.
FUNCTION.
SERVICES THE INST.

INPUT.
PSH; PUL HAVING FORMAT NO. 5.

OUTPUT.
CHORP POINTS TO SH. AFTER
OP-CODE.

CALLS.
CHORP POINTS TO BREAK.
LENGTH,SPACE GET.

CALLED-BY.
SPACE.

GLOBAL-VARIABLES-USED.
SPACE,ABC,SPACE.

GLOBAL-VARIABLES-CHANGED.
LENGTH,SPACE.

ERROR-BITS-SET.
10(F).

SPACE FOR SAVING ACC CONTENTS.
ACCS6:  8,1
ACCS6:  8,1
ACCS6:  8,1
ACCS6:  8,1
ACCS6:  8,1

CONSTANTS.
ENTRY, SAVE ACC CONTENTS.

SET LENGTH

CHECK WHETHER ABFLG IS SET.

ABFLG NOT SET, SEE WHETHER AOR B IS PRESENT.

ERROR, SET ERROR BIT NO 11
NAME: CPX67.


INPUT: CURR POINTS TO CH. AFTER OP-CODE.

OUTPUT: CURR POINTS TO BREAK.

CALLS: STC4,1KH39,INH01,1CI00.

CALLED-BY: SPM25.

GLOBAL-VARIABLES-USED: NSPCE,BSVAL.

GLOBAL-VARIABLES-CHANGED: LENTH,NSPCE.

ERROR-BITS-SET: NONE.

SPACE FOR SAVING ACC CONTENTS.

ACC671: BLK 1
ACC1671: BLK 1
; SAVE ACC CONTENTS, ENTRY
CP167: STA 1,AC1567
STA 1,AC1567
STA 2,AC267
STA 3,AC367

; CHECK IF ADFLG = 1
JSEP ST304
JMP EN167
SUB 1,1

; CHECK WHICH MODE OF ADDRESSING,
JSEP IMM33
JMP IMM67
JSEP IMM31
JMP IMM67
JSEP DIR66
JMP EXT67
JMP DIS67
JMP DIS67
LDA 3,NSPCE
LDA 3,NSPCE
LDA 2,SK67
LDA 2,SK67
STA 2,SK67
STA 2,SK67
LDA 1,NSPCE
AND 1,NSPCE
AND 1,NSPCE
ISZ LENT1
JMP NXT67

EXT67: INC 1,1
INC 1,1
INC 1,1

FORM FIRST BYTE
NXT67: 1,1
1,1
RETURN, AFTER RESTORING ACES.

ENTRY   LIA 1, AC067
         LIA 1, AC167
         LIA 2, AC267
         JRFS AC167

NAME.     STC6c.
FUNCTION. SERVICES THE INSTRUCTION. STS
          STX HAVING FORMAT NO. 5.
INPUT.    OP-CODE TO WA. AFTER
OUTPUT.   OP-CODE TO BREAK.
          LENGTH, NSPACE SET.
CALLS.    STMS, Inc1, Clou.
CALLED BY. SPH29.
GLOBAL-VARIABLES-USED. SAVAL, NSPACE.
GLOBAL-VARIABLES-CHANGED. NSPACE.
ERROR-BITS-SET. 0.

SPACE FOR SAVE OF ACC CONTENTS, ENTRY
     AC1681  . BLK  1
     AC1681  . BLK  1
     AC1681  . BLK  1
     AC1681  . BLK  1
     AC1681  . BLK  1
SAVE ACC CONTENTS, ENTRY.
ST16:1 STA 0,AC.68
STA 1,AC.68
STA 2,AC.68
STA 3,AC.68

SEE WHETHER APLG = 3
JPCS STA6
JMP ENG68

JNE 1,1

CHECK WHICH MODE OF ADDRESSING.
JPCS IN361
INJ 0,2,SN
JMP IN363
JPCS DRT62
JMP EXT68
JMP DRT68

EXT68: INC 1,1
INC68: INC 1,1

FORM FIRST BYTE.
DRT68: MOVLI 1,1
MOVLI 1,1
MOVLI 1,1
MOVLI 1,1

LDA 0,3VAL
ADDS 0,1
LOADS 0,4SPCE
ADD 1,0
STAS 0,4SPCE

RETURN AFTER RESTORING ACCS.
ENG68: LDA 0,AC68
LDA 1, AC169
LDA 2, AC269
JMP < AC369

NAME: J.P55.
FUNCTION: SERVICES THE INSTR. J.P.
JOB HAVING FORMAT NO.
INPUT: CHAIN POINTS TO ON, AFTER
OP-CODE.
OUTPUT: CHAIN POINTS TO BREAK,
LENGTH, NSPACE SET.
CALLS: STB41, INO1, EXOJ2.
CALLED BY: SML25.
GLOBAL-VARIABLES-USED: SVAL, NSPACE.
GLOBAL-VARIABLES-CHANGED: NSPC1.
ERROR-BITS-SET: NONE.

SPACE FOR SAVING ACC CONTENTS:
AC169: BLK 1
AC269: BLK 1
AC369: BLK 1

ENTRY, SAVE ACC CONTENTS:
JM169: STA 0, ACURY
STA 1, AC169
STA 2, AC269
STA 3, AC369

CHECK AFLAG AND SET CHPTR.
```assembly
JSR# ST004
JMP# EN369

; CHECK WHICH MODE
JSR# INQ
JMP# EN369
JSR# EX862
INC# 1,1

; FORM FIRST BYTE
INDEX# MOVL# 1,1
MOVL? 1,1
MOVL? 1,1
LOA# 0,3SVAL
ADCS# 0,1
ADIL# 1,0
STAS# 0,NSPCE
ENDSB# LOA# 0,AC169
LOA# 1,AC169
LOA# 2,AC169
JMP# AC169
```

**NAME:**

**FUNCTION:**

ST34.

**INPUT:**

CHECKS WHETHER ASFLC = 3.

**OUTPUT:**

SHIF POINTS TO C0H. AFTER NP-CODE.

**CALLS:**

CUTA POINTS TO BEG.

OF EXPRESSION.

AC 3 RETURN IF ERROR.

AC ? + 1 RETURN OTHERWISE.

**CALLED BY:**

SUP23, ENR34.

---

224
GLOBAL-VARIABLES-USED.
    LUT.
GLOBAL-VARIABLES-CHANGED.
    ABFLG.
ERROR-BITS-SET.
    ONE.

SPACE FOR SAVING ACC CONTENTS.
ACC4:  3
ACC6:  12

CONSTANTS.
TH34:  3
TH54:  12

SAVE ACC ENTRY
SIEC4:  STA  0,A3804
        STA  3,A3804

CHECK ABFLGS
    LOA  3,ABFLG
    LOA  3,TH504
    SURE  U3,S24
    JMP  EN304

SET CHTK AND RETURN
    JSK  SC223
    JMP  EN304
    JSZ  A3804
    JMP  EN304

SET BIT 1j
EIPC4:  LOA  0,TH504
        STA  0,A3804
        JSK  EH236
        ABFL4:  3

RETURN AFTER RESTORING ACCs.
NAME:  FCG55  
FUNCTION:  SERVICE THE PSEUDO-OP  
INPUT:  CHAIN POINTS TO OR .  
OUTPUT:  SPACE CLEARED, LENGTH SET TO  
          ZERO. LUCTR UPDATED.  
CALLS:  GCH51, GCH57, GCH15,  
        ERE26, SCP23, DEC31.  
CALLED-BY:  SP: 25.  
GLOBAL-VARIABLES-USED:  P-FLG, SPASS.  
GLOBAL-VARIABLES-CHANGED:  CHTF, LENGTH, SPASS, LUCTR.  
ALP-BITS-SET:  2(0), 0

SPACE FOR SAVING ACC CONTENTS

| ACC65 | 4LX   | 1   |  |
| AC155 | 4LX   | 1   |  |
| ACC55 | 4LX   | 1   |  |
| AC755 | 4LX   | 1   |  |

CONSTANTS

| CHF65 | 4LX | 1   | CHFMA | |
| CHL65 | 4LX | 154 | CH9   | |
| CHG65 | 8LX | 54  | CH9   | |
| CHZ65 | 8LX | 51  | CH9   | |
| CH65  | 8LX | 71  | CH9   | |
| CST65 | 8LX | 121 | DEC256| |
| CX65  | 8LX | 40  |       | |
THROSG:
THRES:

ENTRY SAVE ACCE.

FE155:
STA 14,0155
STA 14,0155
STA 34,0255
STA 34,0355

SET CHPT:

JSRS SCR23
JMP END55 ;  ERROR RETURN.

CHPT: SET SAVE VALUE AND CHECK WHICH TYPE OF FLC.

LDA 0,CHTR
STA 0,CHTR
JSRS COMB1
STA 0,JEL55

CHECK WHETHER CH IS DECIMAL

LDA 0,CHTR
LDA 1,CHTR
LJCE 2,CHTR
LJCE 0,ECHTR
JMP LCP55 ; NOT DEC.

CH IS DECIMAL: EVALUATE AND CHECK IF BREAK IS 7,7.

JSZ CHTRP
JSRS DECO1

ACQ 1 CONTAINS VALUE, ACQU CONTAINS BREAK.

CHECK WHETHER BREAK IS 7,7.

LDA 2,020,55
SUBE 0,020,57
JMP LCP55 ; NOT OUTPUT.

VALUE IS IN ACQ 1.
CHECK WHETHER VALUE IS LESS THAN 25%. 
LOAD 0,CH55
SUBLE 3,CHNC
JMP V0155

GET CHARACTER AND PUNCH IT OUT

NEG 1,1
L0V 1,1,SNK
JMP EN055
JCH 0,0,0
STIS 0,0,SPACE
S105 0,LEN
J05S 0,11
S15K 0,0
STIS 0,0,SPACE
S15 0,LEN
L15 0,LOCTA
INC 0,0
STA 1,1
JMP LEL55

DELIMITER IS IN CEL55, CHPTA VALUE IS IN CHP55.
PUNCH IT OUT.

LCP551: LOAD 1,CEL55
LOAD 0,CHP55
STA 0,CHPTA

NXT551 JCP 0,CHPTA
LOAD 3,CHPTA
LOAD 2,CHST55
SUBE 3,2,SNK
JMP ERR55

CHECK IF DELIMITER.

SUBE 0,1,SNK
JMP EN055

CHARACTER IS NOT DELIMITER, PUNCH IT OUT.

JUVS 0,0
STA 0,SPACE
IS2 LETH
JR#10 S1N15

SET LETH AND NSPCE TO U
    STA U, LETH
    STA U, NSPCE
    LDA U, LOC'T
    STA U, LOC'T
    JMA NXT55

SET BIT NO. 3.
    V6155:   LDA U, TH55
              STA U, A6155
              JMA EME25
    &6155:   JBLK 1
              JMA END55

ERROR . SET BIT NO 2
    E8155:   LDA U, TW055
              STA U, ARG55
    &8555:   JBLK 1

STORE ACC AND RETURN
    EN554:   LDA U, AC055
              LDA 1, A6155
              JMA AC355

NAME:
FUNCTION:  F354.
          THIS ROUTINE SERVICE THE
          PENDING OP FOR.
INPUT:    CHIP POINTS TO CH.
          AFTER OP-CODE.
OUTPUT:
LISP POINTS TO BREAK.
LENTH ,SPACE SET TO J.

CALLS:
STEP4,EVL97,GIN15.

CALLED-BY:
SFN25.

GLOBAL-VARIABLES-USED:
LENTH,REP4.

GLOBAL-VARIABLES-CHANGED:
LENTH,REP4,LOCTR.

ERROR-JETS-SET:
NONE.

SPACE FOR SAVING ACC CONTENTS:
AC1641 ,BLK 1
AC1641 ,BLK 1
AC2641 ,BLK 1
AC2641 ,BLK 1

CONSTANTS
O11641 1
O11641 1
O11641 1
O11641 1

ENTRY,SAVE ACC CONTENTS
FC1641 STA 0,AC1654
STA 1,AC1654
STA 2,AC1654

SET CHKTR
JMP ST134
JMP EN354

EVALUATE EXPRESSION:
LEL54: J0R5 EVL97

EXP HAS BEEN EVALUATED. STORE IN SPACE
LSA 0,BREAK


ENV54:  LOA  J, ACH54
       LOA  J, ACH54
       LOA  J, ACH54
       JMP  ACH54


-------

NAME:

FUNCTION: THIS ROUTINE SERVICES THE
INPUT.

OUTPUT.

CALLS.

CALLED-BY.

GLOBAL-VARIABLES-USED.

GLOBAL-VARIABLES-CHANGES.

ERROR-BITS-SET.

SPACE FOR SAVING ACC CONTENTS

ACC56: BLK 1
AC165: BLK 1
AC256: BLK 1
AC356: BLK 1

CONSTANTS

CN56: 1
CN56: 34
CN56: BLK 1

ENTRY, SAVE ACCS

FD165: STA 3,AC65
STA 3,AC156
STA 3,AC256
STA 3,AC356

SET CHPTK

JUB ST304
JUB EN356

EVALUATE EXPRESSION

LEL56: JUB EVL97
EXP HAS BEEN EVALUATED, STORE IN WSPACE

LOAD J, BREAK
STORE J, RKK56
STORE 1, WSPACE

INCREMENT LENGTH, SET LOCTR
STORE 1, LENGTH
STORE LENGTH

CALL ON ROUTINE TO PUNCH IT OUT
JUMP 31115

SET LENGTH TO 1, CLEAR WSPACE
STORE J, 1
STORE 1, LENGTH
STORE 1, WSPACE
STORE 1, LOCTR
STORE 1, 1
STORE 1, LOCTR

CHECK ON BREAK CHARACTER.
LOAD J, RKK56
LOAD 1, C0056
SUB 1, 1
SUBR
JUMP EN056

CHARACTER IS A BREAK, EVALUATE NEXT EXP
STORE J, CHPT
JUMP LBL56

RESTORE ACC CONTENTS AND RETURN.
EN056: LOAD J, ACC56
LOAD 1, AC156
LOAD 2, AC256
JUMP AC356
NAME. BDX46.

FUNCTION. CONVERTS A BINARY NO. INTO A SERIES OF ASCII DECIMAL CHARACTERS.

INPUT. BINARY NO. IN ACC J.

OUTPUT. ASCII CH. ARE INSERTED INTO PTLN STARTING FROM PTR.

CALLS. EPT33.

CALLED-BY. EPG60, SGQ23, LER17.

GLOBAL-VARIABLES-USED. MO.

GLOBAL-VARIABLES-CHANGED. PNR.

ERROR-FLAGS-SET. NONE.

SPACE FOR SAVING ACC CONTENTS.

AC2441  .BLK  1
AC2461  .BLK  1
AC3451  .BLK  1
AC346  .BLK  1

POWERS OF TEN

10 100 1000 10000

100 1000 10000

PILL46  .RDX PTH46
TEMP46  .BLK  1
ZEF46  .BLK  1

SAVE ACCS, ENTFY

234
BC1461 STA U,AC746
STA 1,AC146
STA 2,AC146
STA 3,AC346

SET UP ADDRESS IN TEMP

LDA 3,CTA46
STA TEMP

GET NEXT DIGIT AND GIVE IT OUT

LDA 4, TEMP
ADC 1, SNR
JMP EN346

LDA 0, ZER46

ACC 2 CONTAINS RUNNING VALUE, ACC 0 CONTAINS COUNT

LEL46 STA 1, S7C
ADC 1, SKP
JMP LEL46

CH, IS IN ACC 0, GIVE IT OUT

JSR 1, PRT56
JSZ PRT52
JSZ TEMP
JMP LCP46

STORE ACC AND RETURN

END46 LDA 4, AC746
LDA 1, AC146
LDA 2, AC346
JMP 16, AC346

HERE.
FUNCTION.

INPUT.

OUTPUT.

CALLS.

CALLED-TO.

GLOBAL-VARIABLES-USED.

GLOBAL-VARIABLES-CHANGED.

ENTRY-POINT.

ENTRY-POINT, SAVE ACCS.

SET CHTPE AND ENTER

JMP  ER057

JMP  SCP23
```
LD2 C,1567
STA U,CT,57
DS2 DPTE
LDA 0,NAME

ENTRY INTO NAME

LBL071 JBEU GNC67
BGEJ U,1
JBET GNC67
ADC U,1
INC 2,1
DS2 C,1567
JMP LEL57

RESTORE ACCS AND RETURN

END571 LDA 0,ACU57
LDA 1,ACU57
LDA 2,ACU57
JMP AC57

NAME.
FUNCTION.
ENTER WORD VALUE AFTER CONVERSION INTO HEX INTO LOCATIONS 15-24 OF FRHLN.

INPUT.
NONE.

OUTPUT.
NONE.

CALLS.
W1034, JXX47, EPT03.

CALLED-BY.
W1016.

GLOBAL-VARIABLES-USED.
LENGTH, NAME, VLFLG, VALUE.

GLOBAL-VARIABLES-CHANGED.
APTR.

ERROR-BITS-SET.
```

237
SPACE FOR SAVING ACC CONTENTS.
AC011  3LK  1
AC111  3LK  1
AC211  3LK  1
AC311  3LK  1

CONSTANTS.
CH011  22  ; DEC 16
CH111  27
MX111  377
ONE111  1

SAVE ACC CONTENTS ,ENTRY POINT.
EV111:  STA  0,AC011
       STA  1,AC111
       STA  2,AC211
       STA  3,AC311

SET UP PRPT2 VALUE
LDA  0,CH011
STA  0,PRPT2

CHECK VALUE OF LENTH
LDA  0,LENTH
CPY  0,JSR
JMP  LENTH1 ; LENTH IS 0

LENTH IS NOT ZERO, SET UP ADDRESS IN ACC 2, DETERMINE LENTH VALUE.
LDA  2,NSPCE
LDA  1,ONE11
SUB  1,JSR
JMP  LENTH1 ; LENTH IS 1
SUB  1,JSR
JMP  LENTH1 ; LENTH IS 2
LENGTH IS 5.
ADD 2 OUTSIDE ADDRESS OF BYTES.
PRINT OUT FIRST TWO BYTES.

LDA 0, J, 2
JMP CVT 64
LDA 0, EN 131
STA 0, PRTPR
INC 2, 2

PRINT OUT BYTE 1 OR 3
LON 31:
LDA 0, J, 2
JMP PH F 47
2 NL 31:
JMP 1
2 NL 31:
JMP 1

MOST SIGNIFICANT HALF IS 1000.

LDA L, AG 131
125 G L, 1
LDA 1, MSK 31
AND 1, 10
JMP PPT 3
ISZ PRPTR

LDA 0, AG 131
AND 1, 1
JMP PPT 3
JMP EN 031

LENGTH IS 10.
LTH 31:
LDA 0, J, 2
JMP CVT 64
JMP EN 031

LENGTH IS 5, CHECK VALUE FLAG TO SEE IF
VALUE HAS TO BE PRINTED.
LZ 31:
LDA L, LF LG
MOV 0, J, SNF
JMP EN 031

VALUE FLAG IF NON ZERO.
F: STORE ACOS AND RETURN.

ENC311: LDA 0, AC31
LDA 1, AC131
LDA 2, AC231
JMP AC331

NAME.
FUNCTION.
INPUT.
OUTPUT.
CALLS.
CALLED-BY.
GLOBAL-VARIABLES-USED.
GLOBAL-VARIABLES-CHANGED.
ERROR-BITS-SET.
SPACE FOR SAVING ACC CONTENTS
AC331: .BLK 1
AC131: .BLK 1
AC231: .BLK 1
CONSTANTS
EGF26:
$4528
$31450
$3245
$3U50
$41356
0
EGA26: EGF26
ENTRY: SAVE ACOS.

PC1201: STA 0, AC320
ST1 3, AC320

CHECK B0.015 TO SEE IF NCOS.

LDA 0, B0.015
101 0, B0.014
JMP ENR29

PUNCH OUT NULLS.

JSR @ PN12

PUNCH OUT RECORD

LDA 0, B0.0D0
STA 0, B0.0D0
LDA 0, AC09
JSR ANR05

RESTORE ACOS AND RETURN

ENR320: LDA 0, AC320
JMP @ AC320

NAME:

FUNCTION:

INPUT:

CALLS:

CALLED-BY:

GLOBAL-VARIABLES-USED.
GLOBAL-VARIABLES-CHANGED.
ERROR-STATUS-SET.

SAVE ACC, ENTRY
SP1621 STA 3,40062
STA 1,40162
STA 3,40262
STA 3,40362

EVALUATE AFTER SETTING DATA:
JMP EN92

EVALUATE EXP. NUMBER IS INTERPRETED AS A BIN. NO.
JMP EVLY/1
MOV 1,0
JMP EN92

AC ERROR IN EVALUATION, CALL ON ROUTINE TO PRINT OUT ERRORS.
MOV 1,0
JMP PLFUS

SET SCFLAG.
SUB 1,1
STA 1,SCFLAG

RESTORE ACC, RETURN.
END621 LDA 3,40062
LDA 1,40162
LDA 3,40262
NAME:   PEX3.
FUNCTION:  PUNCHES OUT THE STARTS
RECORD WITH 3 WORDS
OF DATA.
INPUT:    NONE.
OUTPUT:   NONE.
CALLS:    WATER, MK12, BMKH47.
CALLED-BY: INT9.
GLOBAL-VARIABLES-USED:  BPLCV, SNFLG, NAME.
GLOBAL-VARIABLES-CHANGED: OUTCV.
ERROR-BITS-SET:   NONE.

SPACE FOR SAVING ACC CONTENTS:

A00C2:  .BLK  1
A10C2:  .BLK  1
A20C2:  .BLK  1
A30C3:  .BLK  1

CONSTANTS AND COUNTERS:

ST203:  31123
        33150
        30160
        30150
MK203:  .BLK  3
CH203:  .BLK  1
MK303:  .BLK  0
SA303:  .ST203
CT303:  .ST203
CHE303: .BLK  1
; 13:20
; 377
; 377
; 1
; 1
; SAVE ACOS, ENTRY
; LDA 0,3,FLG
; BVC 0,SHR
; JMP EN313
; PUNCH OUT NULLS.
; JSRS PNL12.
; SET UP DEVICE CODES.
; LDA 0,3PDV
; STA 0,OUTDV
; SET UP ADDRESSES
; LDA 0,UAME
; STA 0,IA393
; LDA 0,N293
; STA 0,LC693
; LDA 0,GT693
; STA 0,CH393
; LDA 1,HS393
; LDA 2,SHR13
; GET NAME, STORE IT, UPDATE SUM
; LB-03: LDA 0,4693
; MVS 0,3
; STA 0,LC693
; AND 0,J
ADC
LDAX 0, NC903
AND 1, Y
ADD 0, 2
JSZ NC903
JSNZ LC903
JSZ CN903
JMP LB303

ALL WORDS ENTERED; GET CHECKOUT

GCH 2, 2
GCHV 2, 3
JSK < BF47
G23U 2, 2
LDA 2, G23U
LDA 2, G23U
STA 2, G23U

PUNCH OUT

LDA 2, 3A303
JSK < WR786

RESTORE ACCS AND RETURN

ENDC: LDA 1, A303
LDA 1, A303
LDA 2, A303
JMP < A303

NAME.
FUNCTION.
SERVICES THE FOCUS-O 3 PAGES.
PRINTS OUT HEADING AFTER UPDATING PAGE.

INPUT.
NONE.

OUTPUT.
NONE.

CALLS.
NONE.

CALLED-BY.
PFU3, P=4, 2, 30046, 47F36.
GLOBAL VARIABLES USED: SP, 25, FC, 14.

GLOBAL-VARIABLES-CHANGED: LTF1G, FC, FC, FC, FC, FC, FC.

ENTRY-VALUE-SET: OUT1, FC, FC, FC, FC, FC, FC.

NONE.

START FOR SAVING ACC CONTENTS:

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>BLK</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC01</td>
<td>1</td>
</tr>
<tr>
<td>ACC101</td>
<td>1</td>
</tr>
<tr>
<td>ACC201</td>
<td>1</td>
</tr>
<tr>
<td>ACC301</td>
<td>1</td>
</tr>
</tbody>
</table>

CONSTANTS, LOCATORS

<table>
<thead>
<tr>
<th>MESSAGE</th>
<th>TXT</th>
<th>EXP. LINE</th>
<th>LOC.</th>
<th>VALUE</th>
<th>INPUT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEAS01</td>
<td>MEAS02</td>
<td>3030</td>
<td>0512</td>
<td>0123</td>
<td></td>
</tr>
<tr>
<td>BLK601</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td></td>
</tr>
<tr>
<td>C4T01</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td></td>
</tr>
<tr>
<td>NBL601</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td></td>
</tr>
<tr>
<td>NLL601</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td></td>
</tr>
<tr>
<td>NLL601</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td></td>
</tr>
<tr>
<td>SCS01</td>
<td>SCS02</td>
<td>3030</td>
<td>SCS03</td>
<td>SCS04</td>
<td></td>
</tr>
<tr>
<td>PURL601</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td></td>
</tr>
<tr>
<td>PUG601</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td></td>
</tr>
<tr>
<td>PCF601</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td>0123</td>
<td></td>
</tr>
</tbody>
</table>

ENTRY, SAVE ACCS:

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>STA</th>
<th>ACC601</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA1601</td>
<td>STA</td>
<td>0, ACC601</td>
</tr>
<tr>
<td>STA</td>
<td>1, ACC601</td>
<td></td>
</tr>
<tr>
<td>STA</td>
<td>2, ACC601</td>
<td></td>
</tr>
<tr>
<td>STA</td>
<td>3, ACC601</td>
<td></td>
</tr>
</tbody>
</table>

CHECK WHETHER ACC601:

LOA 0, LTF1G
ICL
JMP ENDB
LDA 0,ENDE
ICL
JMP ENDB

PRINT OUT LEDS.

LDA 6,DL36
JSRS FLP60

SET OUTPUT DEVICE CODE

LDA 6,SL60
STA 6,OUT60

CLEAR PRINT LINE

LDA 1,INT60
DLO 1,1
LDA 2,POFLH
LDA 3,BLX60

LBL60: STA 3,9
INC 8,2
JMP 1,1,SZP
JMP LBL60

INTRODUCE NAME

LDA 2,POFLH
LDA 1,1,BX60
AOU 1,1
LDA 3,NAME
LDA 1,CH60
LDS 1,1
LDA 3,0,3
STA 3,0,2
INC 3,2
INC 1,1,SZP
JMP -5

ENTER PAGE

LDA 0,JST60
STA 0,CR60
LDA 1,SPH40
STA 1,PRPTR
LDA 2,PGAS0

LCP60:
LDA J,M+2
JMP EPTF3
JSN 1,PRBMF
JSN 2,CTR04
JMP LCP6J

ENTER PAGE NO.:
LDA F,PGAS0
STA J,M+1
LDA 1,PGAS0
JSR 4,ECC04B

ALL WORDS ARE ENTERED, PRINT OUT
JSR 5,PRT52

LDA J,PUV1E
STA J,PUV1S
JSR 2,NEA60

PRINT OUT LINE FEEDS
LDA E,PLF00
JSR 4,PLF00

SET FLAG SO THAT LINE IS NOT PRINTED OUT
JSR 4,PLF00
STA B,SOFLG

RESTORE ACS AND RETURN.
END:
LDA J,AC150
LDA 1,AC150
JSR 2,AC250
ALSPJ
<table>
<thead>
<tr>
<th>NAME.</th>
<th>REF.17.</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTION.</td>
<td>LISTS OUT THE TOTAL NO. OF ERRORS.</td>
</tr>
<tr>
<td>INPUT.</td>
<td>NONE.</td>
</tr>
<tr>
<td>OUTPUT.</td>
<td>NONE.</td>
</tr>
<tr>
<td>CALLS.</td>
<td>3046, PRT32.</td>
</tr>
<tr>
<td>CALLED-BY.</td>
<td>PASS1, HELP1.</td>
</tr>
<tr>
<td>GLOBAL-VARIABLES-USED.</td>
<td>PTRLI, SLUR1, PRT3, SF113, EIMPT, EF213, EC1PT.</td>
</tr>
<tr>
<td>GLOBAL-VARIABLES-CHANGES.</td>
<td>PTRL1, OUT1, PRT3.</td>
</tr>
<tr>
<td>ERROR-BITS-SET.</td>
<td>NONE.</td>
</tr>
</tbody>
</table>

**SPACE FOR saving ACCS.**

<table>
<thead>
<tr>
<th>ACC171</th>
<th>0, LK 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC1171</td>
<td>0, LK 1</td>
</tr>
<tr>
<td>ACC2171</td>
<td>0, LK 1</td>
</tr>
<tr>
<td>ACC3171</td>
<td>0, LK 1</td>
</tr>
</tbody>
</table>

**MESSAGE**

<table>
<thead>
<tr>
<th>MESS17</th>
<th>52117</th>
</tr>
</thead>
<tbody>
<tr>
<td>MESS11</td>
<td>52117</td>
</tr>
<tr>
<td>MESS12</td>
<td>46140</td>
</tr>
<tr>
<td>MESS13</td>
<td>42522</td>
</tr>
<tr>
<td>MESS14</td>
<td>51117</td>
</tr>
<tr>
<td>MESS15</td>
<td>511123</td>
</tr>
<tr>
<td>MESS16</td>
<td>20446</td>
</tr>
<tr>
<td>MESS17</td>
<td>20456</td>
</tr>
<tr>
<td>MESS18</td>
<td>74</td>
</tr>
<tr>
<td>MESS19</td>
<td>20440</td>
</tr>
<tr>
<td>MESS20</td>
<td>1</td>
</tr>
<tr>
<td>MESS21</td>
<td>7</td>
</tr>
</tbody>
</table>

**SAVE ACCS, ENTRY**
KE171

STA J,AC017
STA 1,AC117
STA 2,AC217
STA 3,AC317

SET DEVICE CODES

LDA J,LOC
STA J,OUT5V

CLEAN TABLE

LDA 2,PRTLA
LDA 0,3LK17
LDA 1,CHT17
IEG 1,1

STA J,j,2
INC 2,2
INC 1,1,SZR
JMP .3

STORE MESSAGE

LDA 2,PRTLA
LDA 3,MEA17
LDA 1,WDO17
IEG 1,1
LDA J,j,1
STA 2,2
INC 3,3
INC 1,1,SZR
JMP .5

GET 40 OF CARDS CONVERT AND ENTER

LDA 6,PSFLG
LDA 1,OHNC17
SUIE 0,1,1SNR
JMP PCN17

PASS OS 2

LDA 1,ERCT2
LDA 6,2ERT2
JMP NXT17

PASS IS 1

PCH17: LDA 1;ER1TB
LDA 0;IEPT

NXT17: SUB 0;
LDA 1;ST17
STA 1;FPTR1
JMP 3;P445
JMP 3;FRT52

RESTORE ACPS AND RETURN

LDA 0;AC017
LDA 1;AC117
LDA 2;AC217
JMP 3;AC317

NAME. PLFS.
FUNCTION. PRINTS OUT CNT AND LFC.
INPUT. ACC 0 CONTAINS THE NO. OF LFC'S TO BE PRINTED.
OUTPUT. NONE.
CALLS. PUT75.
CALLED BY. LOTS.
GLOBAL-VARIABLES-USED. LTLG, SLOTS.
GLOBAL-VARIABLES-CHANGED. OUTCV.
ERROR-BITS-SET. NONE.

SPACE FOR SAVING ACC CONTENTS.

251
ACCESS: BLK 1
AC208: BLK 1
AC310: BLK 1

CONSTANTS
LPE06: 12
LTO08: 15

SAVE ADDR, ENTRY
FLICE: STA 2,55000E
STA 2,65030E

CHECK LTFLE
LJIA 0,41TEFG
LJIF 24,35030E

SET DEVICE CODES
LDA 3,6LOTS
STA 3,9OUTOV

PRINT LF AND CR
LDA 3,5FETJF
LJPS 3,PUT75
LDA 3,4F03B
LDA 3,3CJ9B
LJPS 3,PUT75
LJIF 2,2,532
LJIF.*1

RETURN
ENDOS: LDA 0,10303E
LJPS 0,1303B

LATE,
FUNCTION.
INPUT.
OUTPUT.
CALLS.
CALLED-BY.
GLOBAL-VARIABLES-USED.
GLOBAL-VARIABLES-CHANGED.
ERROR-BITS-SET.
SPACE FOR SAVING ACOS.
AC12: .BLK 1
AC012: .BLK 1
AC312: .BLK 1
CONSTANTS
CST12: 15
LFD12: 12
NL12: 6
ENTRY, SAVE ACOS.
PHA12: STA 3, AC112
STA 2, AC012
STA 3, AC312
CHECK BIFLG
LDA 0, BIFLG
JMP END12
SET OUTPUT DEVICE ADDRESS
LDA J, SPDIDV
STA L, OUTDV
PRINT OUT C\$\$LF

LDA 1,CHR\$12

JSR 5,PUT75

LDA 2,CHR\$312

JSR 1 PUT75

PRINT OUT nulls

LDA 2,\$H\$112

JMP 2,\$H\$312

NC\$ 2,\$2,SIZE

JMP -2

RESTORE AC\$S AND RETURN

END121 LDA 3,AC\$12

LDA 2,AC\$312

JMP\$ AC\$312

NAME

FUNCTION

DESCRIPTION

INPUT

NONE

OUTPUT

NONE

CALLS

PAG04

CALLED BY

PAG04

GLOBAL-VARIABLES-USED

PAG04, PNOIN

GLOBAL-VARIABLES-CHANGED

NONE

ERROR-INITIATED

NONE

SPACE FOR SAVING AC\$ CONTENTS

254
ACU34 : BLK 1
ACI34 : BLK 1
ACG34 : BLK 1
ACZ34 : BLK 1

CONSTANTS, COUNTERS:
CTR 34 : BLK 1
CNT 34 : 04
ONE 34 : 1

ENTRY, SAVE ACUS.
FV134: STA 0,ACU34
       STA 1,AC134
       STA 2,AC234
       STA 3,AC334

CHECK IF FORMATTING IS MESS.
LDA 3,2PFTB
LDA 0,5,3
LDA 1,ONE34
SUBE 0,1,STA
JMP ENDS4

CHECK FIRST TIME THRO.
LDA 0,AC4T
LDA 1,ONE34
SUBE 0,1,0DF
JMP NXT34

FIRST TIME THRO, INITIALISE CTR34.
LDA 0,CNT34
STA 3,CTR34

CHECK IF END OF PAGE.
NXT34 : DSZ CTR34
       JMP ENDS4

PRINT HEADING.
JSES PAGE34
RESET COUNTER.

LOAD 1, OUT.4
STA 11, OUT.34

STORE ACS AND RETURN.
END 341 LOAD 1, AC 341
LOAD 1, AC 141
LOAD 2, AC 234
JMP AC 334

NAME.
FUNCTION.
LISTS OUT THE SYMBOL TABLE.

INPUT.
NONE.

OUTPUT.
NONE.

CALLS.
NONE.

CALLED-BY.
NONE.

GLOBAL-VARIABLES-USED.
LXY 90, HL 46.

GLOBAL-VARIABLES-CHANGED.
NONE.

SPACE FOR SAVING ACS CONTENTS.

AC 18: 13 1
AC 116: 13 1
AC 218: 13 1
AC 318: 13 1

CONSTANTS, COUNTERS.
LEN16: .BLK 1

CTP16: .BLK 1

FD16: .BLK 4

SET16: .BLK 1,000,000

FIN16: .BLK 1

LIN16: .BLK 3

LSP16: .BLK 10

LEN16: .BLK 10

SAVE REG ENTRY

LE1116: STA 0,AC016
         STA 1,AC116
         STA 2,AC216
         STA 3,AC316

CHECK WHETHER LISTING IS MADC.

LDA 0,ATFLG
JCV 0,3,STA
JMP END13

SET DEVICE CODES.

STA 0,SLOT
STA 0,OUTCV

PRINT OUT BLANK LINES.

LDA 0,BLN16
JABS 0,FLFO8

SYMBOL TABLE LISTING IS MADC.

A METHOD WHICH DOES NOT USE EXTRA STORAGE, BUT
USES A LOT OF REDUNDANT COMPARISONS HAS BEEN USED.

SET LEN16 TO SYMBOL TABLE LENGTH.

LDA 0,OFTB
JMP 0,3,3
STA 0,LEN16
SET ALL ZERO ENTRIES TO 1.

LDA $11018
STA $11018

LDA $3, SYMB
LDA $1, SET16

LBE16: BSR $6, J, 3
STA 1, J, 3

LBE26: LDA $2, SYMB
LDA $3, LAD16
STA 0, SET16
LDA $0, SET16
STA $0, J, SPIE12

SUB
STA $5, SPIE12

LBE36: LDA 0, J, 2
SUBE $3, ENIE12

JMP LE116

ALL NULL ENTRIES HAVE BEEN SET TO 1.

DETERMINE MINIMUM ENTRY.
GET OUT OF LOOP IF MINIMUM VALUE IS 1.

LBE16:

LBE26:

LBE36:

ALL VALUES HAVE BEEN INITIALIZED.
ACC 2 CONTAINS ADDRESS OF SYMBOL TABLE.
ACC 3 CONTAINS ADDRESS OF LOCAL SPACE.
GO INTO LOOP TO DETERMINE MINIMUM VALUE.

RESULT IS NOT ZERO. COMPARE ENTRIES.

JMP $3, J, E2C
JMP $4, SPIE12
FIRST NOC MATCHED. CHECK SECOND NOC.

PICK 1

LOAD J, 1, 3
SUBE 1, 3
JMP Z4318

COMPLETE ENTRIES:

SUBE J, 1, 3
JMP NXT16
JMP REP18

FIRST TWO ENTRIES MATCH:

PICK 1

LOAD J, 2, 3
SUBE 1, 3
JMP NXT16

REPLACE MINIMUM VALUE:

PICK 1

STA 2, F.116
LOAD 2, J, 3
STA 2, J, 3
STA 2, J, 3
LOAD 2, J, 3
STA 2, J, 3

MINIMUM VALUE PLACED.

UPDATE NOC, CHECK WHETHER ALL ENTRIES HAVE BEEN CHECKED.

PICK 1

LOAD J, F.116
ADD J, 3
SUB 3, 3
JMP L318

FIN.13 CONTAINS ADDRESS OF MINIMUM VALUE.

LSP13 CONTAINS MINIMUM VALUE.

CHECK WHETHER FIRST NOC IS 1.00000

LOAD LSP13
LOAD 1, J, 3
JMP L318
SUBE 0,1,SNP
JMP OVA13

ALL ENTRIES HAVE NOT BEEN PRINTED OUT.
PRINT OUT THIS ENTRY AND SET F1: SET TO 1.
CALL ON ROUTINE PNS39 TO PRINT OUT THE ENTRY.
PNS39 NEEDS AS INPUT THE ADDRESS OF ENTRY I, IN CC 2.

LDA 2,FIN13
LNS PNS39

SET BIT 1 TO 1.

LDA 0,FIN13
STA 1,SET13
AND 1,0
STA 0,FIN13

GET NEXT MINIMUM VALUE.

JMP LEB13

ALL ENTRIES HAVE BEEN PRINTED OUT.
CLEAR BIT 1 OF ALL ENTRIES.
THE SUBTRACT 100000 FROM ALL ENTRIES.

OVA13:
LDA 0,EN13
STA 0,ST13
LDA 3,SYM1B
LDA 2,PDF16
LDA 1,SET13

LEB13:
LDA 0,/,S
SUB 1,0
STA 0,/,S

ADD 2,3
DEX CTR16
JMP LEB16

RESTORE ADV AND RETURN.

EB13:
LDA J,AC13
LDA 1,AC13
LDA 2,AC13
JMPX 4,313
NAME: PSY35
FUNCTION: PRINTS OUT AN ENTRY OF
THE SYMBOL TABLE.
INPUT: ARG 2 CONTAINS THE STARTING
ADDRESS OF ENTRY.
OUTPUT: NONE.
CALLS: PUT/8, BNX47.
CALLED BY: LEM16.
GLOBAL-VARIABLES-USED: SYMB.
GLOBAL-VARIABLES-CHANGED: NONE.

SPACE FOR SAVING ACC. CONTENTS:
BLK389: BLK 1
AC139: BLK 1
AC239: BLK 1
AC339: BLK 1

CONSTANTS, COUNTERS:
BLK389: 49
AC139: 177
AC239: 3
AC339: 1

ENTRY POINT, SAVE ACC CONTENTS:
STA 0, AC035
STA 1, AC135
STA 2, AC235

261
STA 6,0055

PRINT OUT ENTRY.
ACC 2 CONTAINS STARTING ADDRESS.

LDA 0,0,133
STA 0,0,133
LDA 1,0,133

LE1381 LDA 0,0,0
MVS 0,0
AND 0,0,0,0
LDA 0,0,1LSK39
JNS 775

PRINT OUT SECOND CHARACTER.

LDA 0,0,2
AND 0,0,0,0
LDA 0,0,1LSK39
JNS 775

INC JSZ 2,0,2
JAP LE133

ALL WORDS PRINTED OUT.
PRINT OUT BLANKS.

LDA 0,0,3LSK39
LDA 1,0,3LSK39
STA 1,0,CTK39
JNS 775
JAP -2

ALL BLANKS PRINTED OUT.
CONVERT LOC. VALUE INTO HEX. INC PRINT OUT.

LDA 2,0,C239
LDA 1,0,4
JNS 3K47
A1331 *BLK 1
A2391 *BLK 1

262
PRINT OUT 4 CHARACTERS.

LDA J, AC 135
LDS J, J
JMPS PLT 75
JMPS PLT 75

PRINT OUT CR, LF.

LDA L, CR 135
JMPS PUT 75
LDA J, LC 135
JMPS PUT 75

ALL CODES PRINTED OUT.

RESTORE ACRS AND RETURN.

END:

LDA 0, AC 135
LDA 1, AC 135
LDA 2, AC 135
JMPS AC 335


.LCD 12000


TABLES (ARRAYS).
<table>
<thead>
<tr>
<th>SYMBOL TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYMT1: BLK 2000</td>
</tr>
<tr>
<td>RX 10</td>
</tr>
<tr>
<td>OBJECT-CODE BUFFER</td>
</tr>
<tr>
<td>GROU1: BLK 40</td>
</tr>
<tr>
<td>TXT 1</td>
</tr>
<tr>
<td>DEFAULT NAME</td>
</tr>
<tr>
<td>DEFT1: TXT /START /</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DIRECT ADDRESS TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIA1: BLK 50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ERROR TABLE FOR PASS-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR-TABLE FOR PASS-2</td>
</tr>
<tr>
<td>ERT1: BLK 50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INPUT SOURCE-LINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>INEL1: BLK 40</td>
</tr>
</tbody>
</table>

| MACHINE-OPERATION TABLE |
PRINT LINE.

P1L1: 3LK 60

WORKING SPACE, SPACE FOR ASSEMBLING INSTRUCTION.

WSPC1: 3LK 2

:CCX 6
:TXTH 0

STUES FOR OTHER ROUTINES.

OP158: JMP 6,3

END
SAMPLE PROGRAM #1

Exhaustive testing of

Various Instruction Types

and Addressing Modes
START

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
<table>
<thead>
<tr>
<th>ER.</th>
<th>LINE</th>
<th>LOC.</th>
<th>VALUE.</th>
<th>INPUT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>00021</td>
<td>0104</td>
<td>8920</td>
<td>ADC A</td>
<td>#AAA+$10</td>
</tr>
<tr>
<td>00022</td>
<td>0106</td>
<td>C931</td>
<td>ADC B</td>
<td>#1</td>
</tr>
<tr>
<td>00023</td>
<td></td>
<td></td>
<td>**/</td>
<td></td>
</tr>
<tr>
<td>00024</td>
<td></td>
<td></td>
<td>* DIRECT</td>
<td></td>
</tr>
<tr>
<td>00025</td>
<td></td>
<td></td>
<td>**</td>
<td></td>
</tr>
<tr>
<td>00026</td>
<td>0108</td>
<td>9910</td>
<td>ADC A</td>
<td>$10</td>
</tr>
<tr>
<td>00027</td>
<td>010A</td>
<td>D910</td>
<td>ADC B</td>
<td>AAA</td>
</tr>
<tr>
<td>00028</td>
<td>010C</td>
<td>990B</td>
<td>ADC A</td>
<td>AAA-$5</td>
</tr>
<tr>
<td>00029</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>00030</td>
<td></td>
<td></td>
<td>* EXTENDED</td>
<td></td>
</tr>
<tr>
<td>00031</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>00032</td>
<td>010E</td>
<td>F901 00</td>
<td>ADC B</td>
<td>$100</td>
</tr>
<tr>
<td>00033</td>
<td>0111</td>
<td>B9FF AB</td>
<td>ADC A</td>
<td>CCC</td>
</tr>
<tr>
<td>00034</td>
<td>0114</td>
<td>F9FF 00</td>
<td>ADC B</td>
<td>CCC-$AB</td>
</tr>
<tr>
<td>00035</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>00036</td>
<td></td>
<td></td>
<td>* INDEXED</td>
<td></td>
</tr>
<tr>
<td>00037</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>00038</td>
<td>0117</td>
<td>A900</td>
<td>ADC A</td>
<td>X</td>
</tr>
<tr>
<td>00039</td>
<td>0119</td>
<td>E900</td>
<td>ADC B</td>
<td>,X</td>
</tr>
<tr>
<td>00040</td>
<td>011B</td>
<td>A910</td>
<td>ADC A</td>
<td>$10,X</td>
</tr>
<tr>
<td>ER. LINE</td>
<td>LOC.</td>
<td>VALUE.</td>
<td>INPUT.</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>------</td>
<td>--------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>00041</td>
<td>011D</td>
<td>E910</td>
<td>ADC B  AAA,X</td>
<td></td>
</tr>
<tr>
<td>00042</td>
<td>011F</td>
<td>A920</td>
<td>ADC A  AAA*$2,X</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00043</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00044</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00045</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00046</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00047</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00048</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00049</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00050</td>
<td>0121</td>
<td>9710</td>
<td>STA A  $10</td>
<td></td>
</tr>
<tr>
<td>00051</td>
<td>0123</td>
<td>D710</td>
<td>STA B  AAA</td>
<td></td>
</tr>
<tr>
<td>00052</td>
<td>0125</td>
<td>970B</td>
<td>STA A  AAA-$5</td>
<td></td>
</tr>
<tr>
<td>00053</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00054</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00055</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00056</td>
<td>0127</td>
<td>F701 00</td>
<td>STA B  $100</td>
<td></td>
</tr>
<tr>
<td>00057</td>
<td>012A</td>
<td>B7FF AB</td>
<td>STA A  CCC</td>
<td></td>
</tr>
<tr>
<td>00058</td>
<td>012D</td>
<td>F7FF 00</td>
<td>STA B  CCC-$AB</td>
<td></td>
</tr>
<tr>
<td>00059</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00060</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* FORMAT NO. 1
* DIRECT
* EXTENDED
* INDEXED
<table>
<thead>
<tr>
<th>Line</th>
<th>Address</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00061</td>
<td>0130 A700</td>
<td>STA A</td>
<td>X</td>
</tr>
<tr>
<td>00063</td>
<td>0132 E700</td>
<td>STA B</td>
<td>,X</td>
</tr>
<tr>
<td>00064</td>
<td>0134 A710</td>
<td>STA A</td>
<td>$10,X</td>
</tr>
<tr>
<td>00065</td>
<td>0136 E710</td>
<td>STA B</td>
<td>AAA,X</td>
</tr>
<tr>
<td>00066</td>
<td>0138 A708</td>
<td>STA A</td>
<td>AAA/$2,X</td>
</tr>
<tr>
<td>00067</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00068</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00069</td>
<td></td>
<td></td>
<td>* FORMAT NO. 2</td>
</tr>
<tr>
<td>00070</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00071</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00072</td>
<td></td>
<td></td>
<td>* ACC A,B</td>
</tr>
<tr>
<td>00073</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00074</td>
<td>013A 48</td>
<td></td>
<td>ASL A</td>
</tr>
<tr>
<td>00075</td>
<td>013B 58</td>
<td></td>
<td>ASL B</td>
</tr>
<tr>
<td>00076</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00077</td>
<td></td>
<td></td>
<td>* EXTENDED</td>
</tr>
<tr>
<td>00078</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>00079</td>
<td>013C 7801 00</td>
<td>ASL</td>
<td>$100</td>
</tr>
<tr>
<td>00080</td>
<td>013F 78FF AB</td>
<td>ASL</td>
<td>CCC</td>
</tr>
<tr>
<td>ER.</td>
<td>LINE.</td>
<td>LOC.</td>
<td>VALUE.</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>00081</td>
<td>0142</td>
<td>78FF 00</td>
<td>ASL CCC-$AB</td>
</tr>
<tr>
<td>00082</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>00083</td>
<td></td>
<td></td>
<td>* INDEXED</td>
</tr>
<tr>
<td>00084</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>00085</td>
<td>0145</td>
<td>6800</td>
<td>ASL X</td>
</tr>
<tr>
<td>00086</td>
<td>0147</td>
<td>6800</td>
<td>ASL ,X</td>
</tr>
<tr>
<td>00087</td>
<td>0149</td>
<td>6810</td>
<td>ASL $10,X</td>
</tr>
<tr>
<td>00088</td>
<td>014B</td>
<td>6810</td>
<td>ASL AAA,X</td>
</tr>
<tr>
<td>00089</td>
<td>014D</td>
<td>6812</td>
<td>ASL AAA+$2,X</td>
</tr>
<tr>
<td>00090</td>
<td></td>
<td></td>
<td>**</td>
</tr>
<tr>
<td>00091</td>
<td></td>
<td></td>
<td>* FORMAT NO. 3</td>
</tr>
<tr>
<td>00092</td>
<td></td>
<td></td>
<td>**</td>
</tr>
<tr>
<td>00093</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>00094</td>
<td></td>
<td></td>
<td>* ACC A,B</td>
</tr>
<tr>
<td>00095</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>00096</td>
<td>014F</td>
<td>36</td>
<td>PSH A</td>
</tr>
<tr>
<td>00097</td>
<td>0150</td>
<td>37</td>
<td>PSH B</td>
</tr>
<tr>
<td>00098</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>00099</td>
<td></td>
<td></td>
<td>* FORMAT NO. 4</td>
</tr>
<tr>
<td>00100</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>ER.</td>
<td>LINE.</td>
<td>LOC.</td>
<td>VALUE.</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td>00101</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00102</td>
<td>* IMMEDIATE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00103</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00104 0151 8C01 00</td>
<td>CPX</td>
<td>#$100</td>
<td></td>
</tr>
<tr>
<td>00105 0154 8CFF AB</td>
<td>CPX</td>
<td>#CCC</td>
<td></td>
</tr>
<tr>
<td>00106 0157 8C00 10</td>
<td>CPX</td>
<td>#AAA</td>
<td></td>
</tr>
<tr>
<td>00107 015A 8CFF 00</td>
<td>CPX</td>
<td>#CCC-$AB</td>
<td></td>
</tr>
<tr>
<td>00108 015D 8C31 20</td>
<td>CPX</td>
<td>*1</td>
<td></td>
</tr>
<tr>
<td>00109</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00110</td>
<td>* DIRECT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00111</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00112 0160 9C10</td>
<td>CPX</td>
<td>$10</td>
<td></td>
</tr>
<tr>
<td>00113 0162 9C10</td>
<td>CPX</td>
<td>AAA</td>
<td></td>
</tr>
<tr>
<td>00114 0164 9C12</td>
<td>CPX</td>
<td>AAA+$2</td>
<td></td>
</tr>
<tr>
<td>00115</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00116</td>
<td>* EXTENDED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00117</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>00118 0166 BC01 00</td>
<td>CPX</td>
<td>$100</td>
<td></td>
</tr>
<tr>
<td>00119 0169 BCFF AB</td>
<td>CPX</td>
<td>CCC</td>
<td></td>
</tr>
<tr>
<td>00120 016C BCFF 00</td>
<td>CPX</td>
<td>CCC-$AB</td>
<td></td>
</tr>
</tbody>
</table>
START

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

ER. LINE. LOC.  VALUE.  INPUT.

00141  * 
00142 017F BFO1 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

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         *
<table>
<thead>
<tr>
<th>ER.</th>
<th>LINE.</th>
<th>LOC.</th>
<th>VALUE.</th>
<th>INPUT.</th>
</tr>
</thead>
<tbody>
<tr>
<td>00181</td>
<td>0306</td>
<td>01</td>
<td></td>
<td>NOP</td>
</tr>
<tr>
<td>00182</td>
<td>0307</td>
<td>19</td>
<td></td>
<td>DAA</td>
</tr>
<tr>
<td>00183</td>
<td>0308</td>
<td>8910</td>
<td></td>
<td>ADCA</td>
</tr>
<tr>
<td>00184</td>
<td>030A</td>
<td>D910</td>
<td></td>
<td>ADCB</td>
</tr>
<tr>
<td>00185</td>
<td>030C</td>
<td>36</td>
<td></td>
<td>PSHA</td>
</tr>
<tr>
<td>00186</td>
<td>030D</td>
<td>58</td>
<td></td>
<td>ASLB</td>
</tr>
<tr>
<td>00190</td>
<td>030E</td>
<td>9903</td>
<td></td>
<td>ADC A</td>
</tr>
<tr>
<td>00191</td>
<td>0310</td>
<td>9909</td>
<td></td>
<td>ADC A</td>
</tr>
<tr>
<td>00192</td>
<td>0312</td>
<td>9911</td>
<td></td>
<td>ADC A</td>
</tr>
<tr>
<td>00193</td>
<td>0314</td>
<td>990B</td>
<td></td>
<td>ADC A</td>
</tr>
<tr>
<td>00194</td>
<td>0316</td>
<td>9905</td>
<td></td>
<td>ADC A</td>
</tr>
<tr>
<td>00195</td>
<td>0318</td>
<td>9901</td>
<td></td>
<td>ADC A</td>
</tr>
<tr>
<td>00196</td>
<td>031A</td>
<td>9901</td>
<td></td>
<td>ADC A</td>
</tr>
<tr>
<td>00197</td>
<td>031C</td>
<td>9909</td>
<td></td>
<td>ADC A</td>
</tr>
<tr>
<td>00200</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>
START

ER. LINE. LOC. VALUE. INPUT.

00201 0200 ORG $200
00202 *
00203 *
00204 0200 PCB $FF,10,,%11
00205 0204 FDB ,F,$FFFF,,%11
00206 *
00207 *
00208 020E PCC /TEXT/
00209 0212 PCC 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
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 *
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

ER. LINE. LOC. VALUE. INPUT.

00041 0010 C60D C60D 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
00048 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 OUT2H 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  

TOTAL ERRORS 00000
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>BADDR</td>
<td>FF62</td>
</tr>
<tr>
<td>COUNT</td>
<td>0004</td>
</tr>
<tr>
<td>CRLF</td>
<td>0010</td>
</tr>
<tr>
<td>GETADR</td>
<td>0047</td>
</tr>
<tr>
<td>GO</td>
<td>0005</td>
</tr>
<tr>
<td>JMONIT</td>
<td>0044</td>
</tr>
<tr>
<td>LSTBYT</td>
<td>0002</td>
</tr>
<tr>
<td>MONIT</td>
<td>FFAB</td>
</tr>
<tr>
<td>NXTBYT</td>
<td>002A</td>
</tr>
<tr>
<td>OUT2H</td>
<td>FF6D</td>
</tr>
<tr>
<td>OUTCH</td>
<td>FF81</td>
</tr>
<tr>
<td>OUTS</td>
<td>FF82</td>
</tr>
<tr>
<td>POLCAT</td>
<td>FF24</td>
</tr>
<tr>
<td>XHI</td>
<td>0000</td>
</tr>
<tr>
<td>XLO</td>
<td>0001</td>
</tr>
</tbody>
</table>
REFERENCES


