IMSAI

SCS

The IMSAI 8080 Monitor, Assembler, and Text Editor, supplied by IMSAI Manufacturing Corporation free of charge, is a modified version of software written by Microtec of Sunnyvale, California for Processor Technology of Berkeley, California who distributed the package free of charge.

IMSAI 8080 SELF-CONTAINED SYSTEM

OPERATING SYSTEM

The IMSAI 8080 Self-Contained System is a software system designed to run on the IMSAI 8080 computer. Included in the package is an Executive to handle memory files, an Assembler, and a line oriented Editor.

To use the system 6K, of memory must be available for use by the system. This memory is allocated as follows:

ØØ4Ø - ØDAB	Operating Program
1ØØØ - 1119	Special System RAM
111A - 17FF	Symbol Table (Assembler Only)

In addition, other memory must be available for source and object files necessary for the user's program.

I/O within the program interacts with I/O ports addressed as follows:

PORT	FUNCTION
2	TTY Data
3	TTY Status Bit Ø indicates TBE Bit 1 indicates DAV
FF	Sense Switch Input ADDRESS - PROGRAMMED INPUT switch seven is used to control file listing.

IMSAI 8080
Self-Contained System
Operating System
Revision 2

Executive Commands

CONTROL-X	Kill current line
ENTR	Enter data to memory
DUMP	Display memory data
FILE	Create, assign or display file information
EXEC	Execute a program
ASSM	Assemble a source file to object code
LIST	List file
DELT	Delete lines of file
1111	Any four numeric digits enters editor
PAGE	Move a page of data
BREK	Set or clear break points
PROC	Proceed from break point
CUST	Optional user command at location 2ØØØ

To initialize the system, start it at ØØØØ. To restart the system without initializing it, start at ØØØ3.

The executive has one error message ....WHAT?.... indicating an improper command or an error on parameters following the command.

Command Format

ENTR AAAA --- Enter data to memory

This command is used to enter data to memory starting at address AAAA and continuing until a slash (/) followed by a carriage return is entered. Data is entered in hexadecimal format.

<u>Example</u>:

```
ENTR 500
0 0A 30 44 FF FE/ (cr)
```

DUMP AAAA BBBB --- Dump contents of memory

This command is used to examine the contents of memory. The values contained in memory from locations AAAA to BBBB are displayed in hexadecimal. Each line of display consists of the contents of up to 16 memory locations. If BBBB is not specified, only locations AAAA will be displayed.

FILE /NAME/ AAAA

This command is used to enter, examine or modify parameters of files created in the system. Up to six files can exist simultaneously with any one of the files "current". Depending on the form of the command, the following parameters the following functions are performed.

FILE /NAME/ AAAA	Create a file with the name, NAME starting at address AAAA and make it current. If a file with the same name already exists, output error message NO NO.
FILE /NAME/ 0	Delete file with name NAME and make no file current. Note: No file can start at location 0.
FILE /NAME/	Get file NAME and make it current. Save all parameters of existing current file.
FILE	Display parameters of the "current" file in the following format with AAAA and BBBB being the beginning of file and end of file addresses: NAME AAAA BBBB
FILES	Display the parameters of all files currently saved by the system.

EXEC AAAA----Execute a program.

This command is used to execute a program at address AAAA.

LIST N----List file

This command is used to display the lines entered by the user into the file. The output consists of the lines in the file starting at line number N. If N is not specified, the display starts at the beginning of the file. The user can terminate the display by raising ADDRESS-PROGRAMMED INPUT switch 7.

DELT L1 L2----Delete line(s) from file

This command is used to delete lines entered by the user from the file. All lines starting at line L1 and continuing up to and including L2 are deleted from the file. If L2 is not specified, only L1 is deleted.

PAGE AAAA BBBB----Move page of data

This command is used to move one page (256 bytes) of data from address AAAA to BBBB.

CUST----Optional user command at location 2000

This command allows any routine to be placed at location 2000 by the user. If the command is terminated by a RET and proper stack operations are used, the system will return in an orderly manner.

BREK or BREK AAAA

This command is used to set or clear break points. If called without the argument AAAA, all break points are cleared.

If called with the argument AAAA, a break point is set at location AAAA. When the break point is encountered in the course of execution, the break point is cleared, all registers are saved, the A register is displayed in the PROGRAMMED OUTPUT on the front panel, the message "AAAA BREAK" is typed and control returns to the executive. The registers are saved in the following locations, and may be examined or modified using the DUMP or ENTR commands.

Location	Register
1000	PSW
1001	A
1002	C
1003	B
1004	E
1005	D
1006	SP (low)
1007	SP (high)
1008	L
1009	H
100A	PC (low)
100B	PC (high)

Restrictions:

(1) A maximum of 8 break points may be set.

(2) Break points may not be set below location 000B.

(3) Setting a break point causes information to be stored into locations 0008-000A, destroying any information already there.

PROC or PROC AAAA

This command is used to proceed from a break point. All registers are restored from the locations specified above, and execution continues from the location specified by the PC, unless the argument AAAA is given, in which case execution begins at location AAAA.

ASSM AAAA BBBB --- Assemble a source file to object code.

This command is used to assemble a source program written by the user and located in the file area. The assembler performs the assembly, assigning addresses to the object code starting at AAAA. On the second pass the object code is placed in memory starting at location BBBB. If BBBB is not specified, it assumes the same value as AAAA. During pass one certain errors are displayed, and during pass two a complete listing is produced.

ASSME AAAA BBBB --- Assemble and list errors only.

This command is the same as ASSM, except that only lines with errors are displayed. Object code is produced just as in ASSM.

TEXT EDITOR

Editor

The editor is a line oriented editor which enables the user to easily create program files in the system. Each line is prefaced by a fixed line number which provides for stable line referencing. Since line numbers can range from 0000 to 9999 (decimal), up to 10,000 lines can exist in each file. As the user types lines on the input device, they are entered into the file area. The editor places all line numbers in sequence and automatically over-writes an existing line in the file, if a new line with the same line number is entered by the user. A feature of the editor is that the file area never contains any wasted space.

Note: The Editor ALWAYS operates on the current file.

The editor does not automatically assign line numbers. The user must first, when entering a line of data, enter a decimal number which will be interpreted as being the line number. Valid line numbers must contain four digits; preceding zeros must be included. An entry to the editor is terminated by the carriate return key. No more than 80 characters may be input for one line.

All lines are ordered by the ascending numeric sequence of their line numbers. If the user wishes to insert lines after the initial entry is made, it is suggested that s/he input the original lines with line numbers at least five units apart.

ASSEMBLER

When the Assembler is given control by the executive, it proceeds to translate the Symbolic 8080 Assembly Language (Source) program into 8080 machine (object) code. The Assembler is a two pass assembler which operates on the "current" file. Features of the Assembler include:

- free format source input.
- symbolic addressing, including forward references and relative symbolic references.
- complex expressions may be used as arguments.
- self defining constants.
- multiple constant forms.
- up to 256 five character symbols.
- reserved names for 8080 registers
- ASCII character code generation
- 6 Pseudo Operations (assembler directives)

The assembler translates those lines contained in the current file into object code. The second character following the line number is considered to be the first source code character position. Hence, the character immediately following the line number should normally be a space. Line numbers are not processed by the assembler; they are merely reproduced on the listing.

The assembler will assemble a source program file composed of STATEMENTS, COMMENTS, and PSEUDO OPERATIONS.

During Pass 1, the assembler allocates all storage necessary for the translated program and defines the values of all symbols used, by creating a symbol table. The storage allocated for the object code will begin at the byte indicated by the 1st parameter in the original Executive ASSM command.

During Pass 2, all expressions, symbols and ASCII constants are evaluated to absolute values and are placed in allocated memory in the appropriate locations. The listing, also produced during Pass 2, indicates exactly what data is in each location of memory.

Statements

Statements may contain either symbolic 8080 machine instructions or pseudo-ops. The structure of such a statement is:

NAME   OPERATION   OPERAND   COMMENT

The name-field, if present, must begin in assembler character position one. The symbol in the name field can contain as many characters as the user wants; however, only the first 5 characters are used in the symbol table to uniquely define a symbol. All symbols in this field must begin with an alphabetic character and may contain no special characters.

The operation field contains either a 8080 operation mnemonic or a system pseudo-operation code.

The operand field contains parameters pertaining to the operation in the operation field. If two arguments are present, they must be separated by a comma. Example:

```
0015 FLOP  MOV M,B  COMMENT
0020 * COMMENT
0025       JMP  BEG
0030       CALL FLOP
0035 BEG   ADI  8+6-4
0040       MOV  A,B
```

All fields are separated and distinguished from one another by the presence of one or more spaces or tabs.

The comment field is for explanatory remarks. It is reproduced on the listing without processing. See example 0015. Comment lines must start with an asterisk (*) in character position 1. See example 0020.

Symbolic Names

To assign a symbolic name to a statement, one merely places the symbol in the name field. To leave off the name field, the user skips two or more spaces after the line number and begins the operation field. If a name is attached to a statement, the assembler assigns it the value of the current Location Counter. The Location Counter always holds the address of the next byte to be assembled. The only exception to this is the EQU pseudo-op. In this case a symbol in the name field is assigned a value which is contained in the operand field of the EQU pseudo-of statement.

Example:

```
0057 POTTS EQU 128
```

assigns the value 128 to the name POTTS. This data can then be used elsewhere in the program, as in ADI POTTS.

Names are defined when they appear in the name field. All defined names may be used as symbolic arguments in the argument field. See examples 0015, 0025, 0030 and 0035.

In addition to user defined names, the assembler has reserved several symbols, the value of which is predetermined. These names may not be used by the user except in the operand field. They are (with their value in parenthesis):

A - the accumulator	(7)
B - Register B	(0)
C - Register C	(1)
D - Register D	(2)
E - Register E	(3)
H - Register H	(4)
L - Register L	(5)
M - Memory (through H,L)	(6)
P - Program Status Word	(6)
S - Stack Pointer	(6)

In addition to the above reserved symbols, there is the single special character symbol ($). This symbol changes in value as the assembly progresses. It is always equated with the value of the program counter after the current instruction is assembled. It may only be used in the operand field.

Examples:

```
JMP  $       means jump to the location
MOV  A,B       after this instruction;
               that is, the MOV instruction

LDA  $+5     means load the data at the
DB   0         fifth location after this
DB   1         location. In this case,
DB   2         the data has the value 5.
DB   3
DB   4
DB   5
```

Relative Symbolic Addressing

If the name of a particular location is known, a nearby location may be specified using the known name and a numeric offset. Example:

```
         JMP    BEG
         JPE    BEG+4
         CC     SUB
         CALL   $+48
     BEG MOV    A,B
         HLT
         MVI    C, 'B'
         INR    B
```

In this example the instruction JMP BEG refers to the MOV A,B instruction. The instruction JPE BEG+4 refers to the INR B instruction. BEG+4 means the address BEG plus four bytes. This form of addressing can be used to locate several bytes before or after a named location.

Constants

The Assembler allows the user to write positive or negative numbers directly in a statement. They will be regarded as decimal constants and their binary equivalents will be used appropriately. All unsigned numbers are considered positive. Decimal constants can be defined using the descriptor "D" after the numeric value. (This is not required, as the default is decimal.)

Hexadecimal constants may be defined using the descriptor "H" after a numeric value. IE. +10H, 10H, 3AH, 0F4H.

Note that a hexadecimal constant cannot start with the digits A-F. In this case, a leading 0 must be included. This enables the assembler to differentiate between a numeric value and a symbol.

ASCII constants may be defined by enclosing the ASCII character within single quote marks, i.e., 'C'. For double word constants, two characters may be defined within one quote string.

Expressions

An expression is a sequence of one or more symbols, constants or other expressions separated by the arithmetic operators plus or minus.

```
PAM +3
ISAB-'A'+52
LOOP+32H-5
```

Expressions are calculated using 16 bit arithmetic. All arithmetic is done modulo 65536. Single byte data cannot contain a value greater than 255 or less than -256. Any value outside this range will result in an assembler error.

Pseudo-Operations

The pseudo-operations are written as ordinary statements, but they direct the assembler to perform certain functions which do not always develop 8080 machine code. The following section describes the pseudo-ops.

ORG----Set Program Origin

Format is

```
label ORG expression
```

where the label is optional but if present will be equaled to the given expression.

END----End of Assembly

The pseudo-op informs the assembler that the last source statement has been read. The assembler will then start on pass 2 and terminate the assembly and pass control back to the executive. This pseudo-op is not needed when assembling from a memory file since the assembler will stop when an end of file indicator has been reached.

EQU----Equal Symbolic Value

Format is

label EQU expression

where label is a symbol the value of which will be determined from the expression, and expression is an expression which when evaluated will be assigned to the symbol given in the name field.

DS----Define Storage

Format is

label DS expression.

The DS causes the assembler to advance the Assembly Program Counter, effectively skipping past a given number of memory bytes.

DB----Define Byte

Format is

label DB expression.

This pseudo-op is used to reserve one byte of storage. The content of the byte is specified in the argument field.

DW----Define Word

This pseudo-op is used to define two bytes of storage. The evaluated argument will be placed in the two bytes; high order 8 bits in the low order byte, and the low order 8 bits in the high order byte. This conforms to the Intel format for two byte addresses.

Assembler Errors

The following error flags are output on the assembler listing when the error occurs. Some of the errors are only output during pass 1.

O	Opcode Error
L	Label Error
D	Duplicate Label Error
M	Missing Label Error
V	Value Error
U	Undefined Symbol
S	Snytax Error
R	Register Error
A	Argument Error.

OBJECT TAPE FORMAT

The IMSAI Self-Contained System is supplied on paper tape in a blocked hexadecimal format. The data on the tape is blocked into discrete records, each record containing record length, record type, memory address and checksum information in addition to data. A frame-by-frame description is as follows:

Frame 0	Record Mark. Signals the start of a record. The ASCII character colon (":" HEX 3A) is used as the record mark.
Frames 1,2 (0-9,A-F)	Record Length. Two ASCII characters representing a hexadecimal number in the range 0 to 'FF' (0 to 255). This is the count of actual data bytes in the record type or checksum. A record length of 0 indicates end of file.
Frames 3 to 6	Load Address. Four ASCII characters that represent the initial memory location where the data following will be loaded. The first data byte is stored in the location pointed to by the load address; succeeding data bytes are loaded into ascending addresses.
Frames 7, 8	Record Type. Two ASCII characters. Currently all records are type 0. This field is reserved for future expansion.
Frames 9 to 9+2* (Record Length) -1	Data. Each 8 bit memory word is represented by two frames containing the ASCII characters (0 to 9, A to F) to represent a hexadecimal value 0 to 'FF'H (0 to 255).

Frames 9+2* (Record Length) to 9+2* (Record Length) +1	Checksum. The checksum is the negative of the sum of all 8 bit bytes in the record since the record mark (":") evaluated modulus 256. That is, if you add together all the 8 bit bytes, ignoring all carries out of an 8-bit sum, then add the checksum, the result is zero.

Example: If memory locations 1 through 3 contain 53F8EC, the format of the hex file produced when these locations are punched is:

```
:0300010053F8ECC5
```

SAVING AND RESTORING PROGRAMS

While the system has no explicit provision for saving and restoring programs, it is possible to do so with an ASR style teletype. The procedure is as follows:

1. Make the file you want to save the current file.
2. Type 'LIST', but don't type the carriage return.
3. Turn on the paper tape punch.
4. Type carriage return. The program will be listed on the teletype and simultaneously punched on the paper tape punch.
5. When the 'LIST' is completed, turn off the punch.

The procedure for restoring the file is as follows:

1. Make the file you want to restore into the current file.
2. Mount the tape in the paper tape reader.
3. Start the paper tape reader. The program will be automatically read in.

An analogous procedure, using the DUMP and ENTR commands, may be used to save and restore object code.

```
                     ; REVISION 2              06 OCT 76
                     ;
                     ; ******** SELF CONTAINED SYSTEM ********
                     ;
0000                         ORG     00H
0000 C34000                  JMP     INITA    ;DEAD START
0003 C36700                  JMP     EOR      ;RESTART MONITOR
                     ;
0006                         ORG     08H
0008 C32E0D                  JMP     BRKP     ;BREAKPOINT RESTART
                     ;
0008                         ORG     40H
                     ;
                     ; THIS ROUTINE SETS UP THE SIO BOARD
                     ;
0040 3EAA            INITA:  MVI     A,0AAH   ;GET DUMMY MODE WORD
0042 D303                    OUT     TTS      ;OUTPUT IT
0044 3E40                    MVI     A,40H    ;GET RESET BIT
0046 D303                    OUT     TTS      ;RESET SIO BOARD
0048 3ECE                    MVI     A,0CEH   ;GET REAL MODE WORD
004A D303                    OUT     TTS      ;SET THE MODE FOR REAL
004C 3E37                    MVI     A,37H    ;GET THE COMMAND
004E D303                    OUT     TTS      ;OUTPUT IT
                     ;
                     ; THIS ROUTINE INITIALIZES THE FILE AREA FOR SUBSEQUENT
                     ; PROCESSING
                     ;
0050 212410                  LXI     H,FILE0
0053 0E4E                    MVI     C,MAXFIL*FELEN
0055 AF                      XRA     A
0056 77              INIT2:  MOV     M,A
0057 23                      INX     H
0058 0D                      DCR     C
0059 C25600                  JNZ     INIT2
                     ;
                     ; CLEAR THE BREAKPOINT TABLE
                     ;
005C 0618                    MVI     B,NBR*3
005E 210C10                  LXI     H,BRT
0061 77              INIT3:  MOV     M,A
0062 23                      INX     H
0063 05                      DCR     B
0064 C26100                  JNZ     INIT3
                     ;
                     ; THIS IS THE STARTING POINT OF THE SELF CONTAINED
                     ; SYSTEM ONCE THE SYSTEM HAS BEEN INITIALIZED.  COMMANDS
                     ; ARE READ FROM THE USER, EXECUTED, AND CONTROL RETURNS
                     ; BACK TO THIS POINT TO READ ANOTHER COMMAND.
                     ;
0067 31B210          EOR:    LXI     SP,AREA+18
006A CD0E01                  CALL    CRLF     ;PRINT C/R, LINE FEED
006D CD8000                  CALL    READ     ;READ INPUT LINE
0070 23                      INX     H
0071 7E                      MOV     A,M      ;FETCH FIRST CHARACTER
0072 FE3A                    CPI     '9'+1    ;COMMAND OR LINE NUMBER?
0074 DAB504                  JC      LINE     ;JUMP IF LINE FOR FILE
0077 CD7301                  CALL    VALC     ;GET COMMAND VALUES
007A CD2B01                  CALL    COMM     ;CHECK LEGAL COMMANDS
007D C36700                  JMP     EOR
                     ;
                     ; THIS ROUTINE READS IN A LINE FROM THE TTY AND PLACES
                     ; IT IN AN INPUT BUFFER.
                     ; THE FOLLOWING ARE SPECIAL CHARACTERS
                     :   CR             TERMINATES READ ROUTINE
```

```
                 ;   LF           NOT RECOGNIZED BY ROUTINE
                 ;   CTRL X       DELETE CURRENT LINE
                 ;   DEL          DELETE CHARACTER
                 ; ALL DISPLAYABLE CHARACTERS BETWEEN BLANK & Z AND THE
                 ; ABOVE ARE RECOGNIZED BY THE READ ROUTINE, ALL OTHERS
                 ; ARE SKIPPED OVER.  THE ROUTINE WILL NOT ACCEPT MORE
                 ; CHARACTERS THAN THE INPUT BUFFER WILL HOLD.
                 ;
0080 21C710      READ:   LXI     H,IBUF  ;GET INPUT BUFFER ADDRESS
0083 227410              SHLD    ADDS    ;SAVE ADDRESS
0086 1E02                MVI     E,2     ;INITIALIZE CHARACTER COUNT
0088 CDF600      NEXT:   CALL    IN8     ;READ A LINE
008B 78                  MOV     A,B
008C FE18                CPI     24      ;CHECK FOR CTRL X
008E C29700              JNZ     CR
0091 CD0E01              CALL    CRLF    ;OUTPUT A CRLF
0094 C38000              JMP     READ
0097 FE0D        CR:     CPI     ASCR    ;GET AN ASCII CR
0099 C2B200              JNZ     DEL
009C 7D                  MOV     A,L
009D FEC7                CPI     IBUF AND 0FFH   ;CHECK FOR FIRST CHAR
009F CA8000              JZ      READ
00A2 360D                MVI     M,ASCR  ;PLACE CR AT END OF LINE
00A4 23                  INX     H
00A5 3601                MVI     M,1     ;PLACE EOF INDICATOR IN LINE
00A7 23                  INX     H
00A8 3E1A                MVI     A,IBUF+83 AND 0FFH
00AA CDE100              CALL    CLER    ;CLEAR REMAINING BUFFER
00AD 21C610              LXI     H,IBUF-1
00B0 73                  MOV     M,E     ;SAVE CHARACTER COUNT
00B1 C9                  RET
00B2 FE7F        DEL:    CPI     127     ;CHECK FOR DELETE CHARACTER
00B4 C2C700              JNZ     CHAR
00B7 3EC7                MVI     A,IBUF AND 0FFH
00B9 BD                  CMP     L       ;IS THIS 1ST CHARACTER
00BA CA8800              JZ      NEXT
00BD 2B                  DCX     H       ;DECREMENT POINTER
00BE 1D                  DCR     E       ;DECREMENT COUNT
00BF 065F        BSPA:   MVI     B,5FH
00C1 CD0301              CALL    OUT8
00C4 C38800              JMP     NEXT
00C7 FE20        CHAR:   CPI     ' '     ;CHECK FOR LEGAL CHARACTER
00C9 DA8800              JC      NEXT
00CC FE5B                CPI     'Z'+1
00CE D28800              JNC     NEXT
00D1 47                  MOV     B,A
00D2 CD0301              CALL    OUT8    ;ECHO CHARACTER
00D5 70                  MOV     M,B
00D6 3E18                MVI     A,IBUF+81 AND 0FFH
00D8 BD                  CMP     L       ;CHECK FOR END OF LINE
00D9 CABF00              JZ      BSPA
00DC 23                  INX     H
00DD 1C                  INR     E       ;INCREMENT CHARACTER COUNT
00DE C38800              JMP     NEXT
                 ;
                 ;
                 ;
                 ; THIS ROUTINE IS USED TO BLANK OUT A PORTION OF MEMORY
                 ;
00E1 BD          CLER:   CMP     L
00E2 C8                  RZ
00E3 3620                MVI     M,' '   ;PLACE BLANK IN MEMORY
00E5 23                  INX     H
00E6 C3E100              JMP     CLER
                 ;
                 ; SEE IF TTY INPUT READY AND CHECK FOR CTRL X.
```

```
                  ;
                  ; RETURN WITH ZERO SET IFF CTRL X SEEN.
                  ;
00E9 DB03         INK:    IN      TTS       ;GET TTY STATUS
00EB 2F                   CMA               ;INVERT STATUS
00EC E602                 ANI     TTYDA     ;IS DATA AVAILABLE?
00EE C0                   RNZ               ;RETURN IF NOT
00EF DB02                 IN      TTI       ;GET THE CHAR
00F1 E67F                 ANI     07FH      ;STRIP OFF PARITY
00F3 FE18                 CPI     'X'-40H   ;IS IT A CTRL X?
00F5 C9                   RET
                  ;
                  ; THIS ROUTINE READS A BYTE OF DATA FROM THE USART
                  ;
00F6 DB03         IN8:    IN      TTS       ;READ USART STATUS
00F8 E602                 ANI     TTYDA
00FA CAF600               JZ      IN8
00FD DB02                 IN      TTI       ;READ DATA
00FF E67F                 ANI     127       ;STRIP OFF PARITY
0101 47                   MOV     B,A
0102 C9                   RET
                  ;
                  ; THIS ROUTINE OUTPUTS A BYTE OF DATA TO THE USART
                  ;
0103 DB03         OUT8:   IN      TTS       ;READ STATUS
0105 E601                 ANI     TTYTR
0107 CA0301               JZ      OUT8
010A 78           OK:     MOV     A,B
010B D302                 OUT     TTO       ;TRANSMIT DATA
010D C9                   RET
                  ;
                  ; THIS ROUTINE WILL OUTPUT A CARRIAGE RETURN AND
                  ; LINE FEED FOLLOWED BY TWO DELETE CHARACTERS WHICH
                  ; PROVIDE TIME FOR PRINT HEAD TO RETURN.
                  ;
010E 060D         CRLF:   MVI     B,13      ;CR
0110 CD0301               CALL    OUT8
0113 060A         LF:     MVI     B,10      ;LF
0115 CD0301               CALL    OUT8
0118 067F                 MVI     B,127
011A CD0301               CALL    OUT8
011D CD0301               CALL    OUT8
0120 C9                   RET
                  ;
                  ; THIS ROUTINE JUMPS TO A LOCATION IN MEMORY GIVEN BY
                  ; THE INPUT COMMAND AND BEGINS EXECUTION OF PROGRAM
                  ; THERE.
                  ;
0121 CD0003       EXEC:   CALL    VCHK      ;CHECK FOR PARAMETER
0124 CD0E01               CALL    CRLF
0127 2A8A10               LHLD    BBUF      ;FETCH ADDRESS
012A E9                   PCHL              ;JUMP TO PROGRAM
                  ;
                  ;
                  ;
                  ;
                  ; THIS ROUTINE CHECKS THE INPUT COMMAND AGAINST ALL
                  ; LEGAL COMMANDS STORED IN A TABLE.  IF A LEGAL COMMAND
                  ; IS FOUND, A JUMP IS MADE TO THAT ROUTINE.  OTHERWISE
                  ; AN ERROR MESSAGE IS OUTPUT TO THE USER.
                  ;
012B 118E02       COMM:   LXI     D,CTAB    ;COMMAND TABLE ADDRESS
012E 060B                 MVI     B,NCOM    ;NUMBER OF COMMANDS
0130 3E04                 MVI     A,4       ;LENGTH OF COMMAND
0132 329510               STA     NCHR      ;SAVE
0135 CD3C01               CALL    COMS      ;SEARCH TABLE
0138 C25A04               JNZ     WHAT      ;JUMP IF ILLEGAL COMMAND
```

```
0138 E9                 PCHL            ;BE HERE NOW
                ;
                ;
                ;
                ;
                ;
                ; THIS ROUTINE CHECKS TO SEE IF A BASE CHARACTER STRING
                ; IS EQUAL TO ANY OF THE STRINGS CONTAINED IN A TABLE
                ; POINTED TO BY D,E.  THE TABLE CONSISTS OF ANY NUMBER
                ; OF CHARS, WITH 2 BYTES CONTAINING VALUES ASSOCIATED
                ; WITH IT.  REG B CONTAINS THE # OF STRINGS TO COMPARE.
                ; THIS ROUTINE CAN BE USED TO SEARCH THROUGH A COMMAND
                ; OR SYMBOL TABLE.  ON RETURN, IF THE ZERO FLAG IS SET,
                ; A MATCH WAS FOUND; IF NOT, NO MATCH WAS FOUND.  IF
                ; A MATCH WAS FOUND, D,E POINT TO THE LAST BYTE
                ; ASSOCIATED WITH THE CHARACTER STRING.  IF NOT, D,E
                ; POINT TO THE NEXT LOCATION AFTER THE END OF THE TABLE.
                ;
013C 2A7410     COMS:   LHLD    ADDS    ;FETCH COMPARE ADDRESS
013F 3A9510             LDA     NCHR    ;GET LENGTH OF STRING
0142 4F                 MOV     C,A
0143 CD5301             CALL    SEAR    ;COMPARE STRINGS
0146 1A                 LDAX    D       ;FETCH VALUE
0147 6F                 MOV     L,A
0148 13                 INX     D
0149 1A                 LDAX    D       ;FETCH VALUE
014A 67                 MOV     H,A
014B C8                 RZ
014C 13                 INX     D       ;SET TO NEXT STRING
014D 05                 DCR     B       ;DECREMENT COUNT
014E C23C01             JNZ     COMS
0151 04                 INR     B       ;CLEAR ZERO FLAG
0152 C9                 RET
                ;
                ;
                ;
                ; THIS ROUTINE CHECKS TO SEE IF TWO CHARACTER STRINGS IN
                ; MEMORY ARE EQUAL.  THE STRINGS ARE POINTED TO BY D,E
                ; AND H,L.  ON RETURN, THE ZERO FLAG SET INDICATES A
                ; MATCH.  REG C INDICATES THE LENGTH OF THE STRINGS. ON
                ; RETURN, THE POINTERS POINT TO THE NEXT ADDRESS AFTER
                ; THE CHARACTER STRINGS.
                ;
0153 1A         SEAR:   LDAX    D       ;FETCH CHARACTER
0154 BE                 CMP     M       ;COMPARE CHARACTERS
0155 C25F01             JNZ     INCA
0158 23                 INX     H
0159 13                 INX     D
015A 0D                 DCR     C       ;DECREMENT CHARACTER COUNT
015B C25301             JNZ     SEAR
015E C9                 RET
015F 13         INCA:   INX     D
0160 0D                 DCR     C
0161 C25F01             JNZ     INCA
0164 0C                 INR     C       ;CLEAR ZERO FLAG
0165 C9                 RET
                ;
                ; THIS ROUTINE ZEROES OUT A BUFFER IN MEMORY WHICH IS
                ; THEN USED BY OTHER SCANNING ROUTINES.
                ;
0166 AF         ZBUF:   XRA     A       ;GET A ZERO
0167 118A10             LXI     D,ABUF+12 ;BUFFER ADDRESS
016A 060C               MVI     B,12    ;BUFFER LENGTH
016C 1B         ZBU1:   DCX     D       ;DECREMENT ADDRESS
016D 12                 STAX    D       ;ZERO BUFFER
016E 05                 DCR     B
```

```
016F C26C01             JNZ     ZBU1
0172 C9                 RET
                ;
                ; THIS ROUTINE CALLS ETRA TO OBTAIN THE INPUT PARAMETER
                ; VALUES AND CALLS AN ERROR ROUTINE IF AN ERROR OCCURRED
                ; IN THAT ROUTINE.
                ;
0173 CD7A01     VALC:   CALL    ETRA    ;GET INPUT PARAMETERS
0176 DA5A04             JC      WHAT    ;JUMP IF ERROR
0179 C9                 RET
                ;
                ; THIS ROUTINE EXTRACTS THE VALUES ASSOCIATED WITH A
                ; COMMAND FROM THE INPUT STREAM AND PLACES THEM IN THE
                ; ASCII BUFFER (ABUF).  IT ALSO CALLS A ROUTINE TO
                ; CONVERT THE ASCII HEXADECIMALS TO BINARY AND STORES
                ; THEM IN THE BINARY BUFFER (BBUF).  ON RETURN, CARRY
                ; SET INDICATES AN ERROR IN INPUT PARAMETERS.
                ;
017A 210000     ETRA:   LXI     H,0     ;GET A ZERO
017D 228C10             SHLD    BBUF+2  ;ZERO VALUE
0180 227610             SHLD    FBUF    ;SET NO FILE NAME
0183 CD6601             CALL    ZBUF    ;ZERO BUFFER
0186 21C610             LXI     H,IBUF-1
0189 23         VAL1:   INX     H
018A 7E                 MOV     A,M     ;FETCH INPUT CHARACTER
018B FE20               CPI     ' '     ;LOOK FOR FIRST CHARACTER
018D 3F                 CMC
018E D0                 RNC             ;RETURN IF NO CARRY
018F C28901             JNZ     VAL1    ;JUMP IF NO BLACK
0192 229610             SHLD    PNTR    ;SAVE POINTER
0195 CD0D09             CALL    SBLK    ;SCAN TO FIRST PARAMETER
0198 3F                 CMC
0199 D0                 RNC             ;RETURN IF CR
019A FE2F               CPI     '/'
019C C2C401             JNZ     VAL5    ;NO FILE NAME
019F 117610             LXI     D,FBUF  ;NAME FOLLOWS PUT IN FBUF
01A2 0E05               MVI     C,NMLEN
01A4 23         VAL2:   INX     H
01A5 7E                 MOV     A,M
01A6 FE2F               CPI     '/'
01A8 CAB401             JZ      VAL3
01AB 0D                 DCR     C
01AC FA5A04             JM      WHAT
01AF 12                 STAX    D       ;STORE FILE NAME
01B0 13                 INX     D
01B1 C3A401             JMP     VAL2
01B4 3E20       VAL3:   MVI     A,' '   ;GET AN ASCII SPACE
01B6 0D         VAL4:   DCR     C
01B7 FABF01             JM      DONE
01BA 12                 STAX    D       ;FILL IN WITH SPACES
01BB 13                 INX     D
01BC C3B601             JMP     VAL4
01BF CD1409     DONE:   CALL    SBL2
01C2 3F                 CMC
01C3 D0                 RNC
01C4 117E10     VAL5:   LXI     D,ABUF
01C7 CD750B             CALL    ALPS    ;PLACE PARAMETER IN BUFFER
01CA 78                 MOV     A,B     ;GET DIGIT COUNT
01CB FE05               CPI     5       ;CHECK NUMBER OF DIGITS
01CD 3F                 CMC
01CE D8                 RC              ;RETURN IF TOO MANY DIGITS
01CF 017E10             LXI     B,ABUF
01D2 CD1802             CALL    AHEX    ;CONVERT VALUE
01D5 D8                 RC              ;ILLEGAL CHARACTER
01D6 228A10             SHLD    BBUF    ;SAVE   IN BINARY BUFFER
01D9 217E10             LXI     H,ABUF
```

```
01DC CD8D05           CALL    NORM     ;NORMALIZE ASCII VALUE
01DF CD0D09           CALL    SBLK     ;SCAN TO NEXT PARAMETER
01E2 3F               CMC
01E3 D0               RNC              ;RETURN IF CR
01E4 118210           LXI     D,ABUF+4
01E7 CD7508           CALL    ALPS     ;PLACE PARAMETER IN BUFFER
01EA 78               MOV     A,B      ;GET DIGIT COUNT
01EB FE05             CPI     5        ;CHECK NUMBER OF DIGITS
01ED 3F               CMC
01EE D8               RC               ;RETURN IF TOO MANY DIGITS
01EF 018210           LXI     B,ABUF+4
01F2 CD1B02           CALL    AHEX     ;CONVERT VALUE
01F5 D8               RC               ;ILLEGAL VALUE
01F6 228C10           SHLD    BBUF+2   ;SAVE IN BINARY BUFFER
01F9 218210           LXI     H,ABUF+4
01FC CD8D05           CALL    NORM     ;NORMALIZE ASCII VALUE
01FF B7               ORA     A        ;CLEAR CARRY
0200 C9               RET
                ;
                ; THIS ROUTINE FETCHES DIGITS FROM THE BUFFER ADDRESSED
                ; BY B,C AND CONVERTS THE ASCII DECIMAL DIGITS INTO
                ; BINARY.  UP TO A 16-BIT VALUE CAN BE CONVERTED.  THE
                ; SCAN STOPS WHEN A BINARY ZERO IS FOUND IN THE BUFFER.
                ;
0201 210000     ADEC: LXI     H,0      ;GEP A 16 BIT ZERO
0204 0A         ADE1: LDAX    B        ;FETCH ASCII DIGIT
0205 B7               ORA     A        ;SET ZERO FLAG
0206 C8               RZ               ;RETURN IFF FINISHED
0207 54               MOV     D,H      ;SAVE CURRENT VALUE
0208 5D               MOV     E,L      ;SAVE CURRENT VALUE
0209 29               DAD     H        ;TIMES TWO
020A 29               DAD     H        ;TIMES TWO
020B 19               DAD     D        ;ADD IN ORIGINAL VALUE
020C 29               DAD     H        ;TIMES TWO
020D D630             SUI     48       ;ASCII BIAS
020F FE0A             CPI     10       ;CHECK FOR LEGAL VALUE
0211 3F               CMC
0212 D8               RC               ;RETURN IF ERROR
0213 5F               MOV     E,A
0214 1600             MVI     D,0
0216 19               DAD     D        ;ADD IN NEXT DIGIT
0217 03               INX     B        ;INCREMENT POINTER
0218 C30402           JMP     ADE1
                ;
                ;
                ; THIS ROUTINE FETCHES DIGITS FROM THE BUFFER ADDRESSED
                ; BY B,C AND CONVERTS THE ASCII HEXADECIMAL DIGITS INTO
                ; BINARY.  UP TO A 16-BIT VALUE CAN BE CONVERTED.  THE
                ; SCAN STOPS WHEN A BINARY ZERO IS FOUND IN THE BUFFER.
                ;
021B 210000     AHEX: LXI     H,0      ;GET A 16 BIT ZERO
021E 0A         AHE1: LDAX    B        ;FETCH ASCII DIGIT
021F B7               ORA     A        ;SET ZERO FLAG
0220 C8               RZ               ;RETURN IF DONE
0221 29               DAD     H        ;LEFT SHIFT
0222 29               DAD     H        ;LEFT SHIFT
0223 29               DAD     H        ;LEFT SHIFT
0224 29               DAD     H        ;LEFT SHIFT
0225 CD3202           CALL    AHS1     ;CONVERT TO BINARY
0228 FE10             CPI     10H      ;CHECK FOR LEGAL VALUE
022A 3F               CMC
022B D8               RC               ;RETURN IF ERROR
022C 85               ADD     L
022D 6F               MOV     L,A
022E 03               INX     B        ;INCREMENT POINTER
022F C31E02           JMP     AHE1
```

```
                    ;
                    ; THIS SUBROUTINE CONVERTS ASCII HEX DIGITS INTO BINARY
                    ;
0232 D630           AHSI:   SUI     48          ;ASCII BIAS
0234 FE0A                   CPI     10          ;DIGIT 0-10
0236 D8                     RC
0237 D607                   SUI     7           ;ALPHA BIAS
0239 C9                     RET
                    ;
                    ;
                    ; THIS ROUTINE CONVERTS A BINARY VALUE TO ASCII
                    ; HEXADECIMAL AND OUTPUTS THE CHARACTERS TO THE TTY.
                    ;
023A CD8602         HOUT:   CALL    BINH        ;CONVERT VALUE
023D 217410                 LXI     H,HCON      ;CONVERSION AREA
0240 46             CHOT:   MOV     B,M         ;FETCH OUTPUT CHARACTER
0241 CD0301                 CALL    OUT8        ;OUTPUT CHARACTER
0244 23                     INX     H
0245 46                     MOV     B,M         ;FETCH CHARACTER
0246 CD0301                 CALL    OUT8        ;OUTPUT CHARACTER
0249 C9                     RET
                    ;
                    ; THIS ROUTINE DOES THE SAME AS ABOVE BUT OUTPUTS A
                    ; BLANK AFTER THE LAST CHARACTER
                    ;
024A CD3A02         HOTB:   CALL    HOUT        ;CONVERT AND OUTPUT
024D CD5D02                 CALL    BLKI        ;OUTPUT A BLANK
0250 C9                     RET
                    ;
                    ;
                    ; THIS ROUTINE CONVERTS A BINARY VALUE TO ASCII
                    ; DECIMAL DIGITS AND OUTPUTS THE CHARACTERS TO THE TTY
                    ;
0251 CDA302         DOUT:   CALL    BIND        ;CONVERT VALUE
0254 CD3D02                 CALL    HOUT+3      ;OUTPUT VALUE (2 DIGITS)
0257 23                     INX     H
0258 46                     MOV     B,M         ;GET LAST DIGIT
0259 CD0301                 CALL    OUT8        ;OUTPUT
025C C9                     RET
                    ;
                    ; THIS ROUTINE OUTPUTS A BLANK
                    ;
025D 0620           BLKI:   MVI     B,' '       ;GET A BLANK
025F CD0301                 CALL    OUT8
0262 C9                     RET
                    ;
                    ;
                    ; THIS ROUTINE IS USED BY OTHER ROUTINES TO INCREMENT
                    ; THE STARTING ADDRESS IN A COMMAND AND COMPARE IT WITH
                    ; THE FINAL ADDRESS IN THE COMMAND.  ON RETURN, THE
                    ; CARRY FLAG SET INDICATES THAT THE FINAL ADDRESS HAS
                    ; BEEN REACHED.
                    ;
0263 2A8A10         ACHK:   LHLD    BBUF        ;FETCH START ADDRESS
0266 3A8D10                 LDA     BBUF+3      ;STOP ADRESS (HIGH)
0269 BC                     CMP     H           ;COMPARE ADDRESSES
026A C27502                 JNZ     ACH1
026D 3A8C10                 LDA     BBUF+2      ;STOP ADDRESS (LOW)
0270 BD                     CMP     L           ;COMPARE ADDRESSES
0271 C27502                 JNZ     ACH1
0274 37                     STC                 ;SET CARRY IF EQUAL
0275 23             ACH1:   INX     H           ;INCREMENT START ADDRESS
0276 228A10                 SHLD    BBUF        ;STORE START ADDRESS
0279 C9                     RET
                    ;
                    ;
```

```
                ;
                ; THIS ROUTINE OUTPUTS CHARACTERS OF A STRING
                ; UNTIL A CARRIAGE RETURN IS FOUND.
                ;
027A 46         SCRN:   MOV     B,M     ;FETCH CHARACTER
027B 3E0D               MVI     A,13    ;CARRIAGE RETURN
027D B8                 CMP     B       ;CHARACTER = CR?
027E C8                 RZ
027F CD0301             CALL    OUT8    ;OUTPUT CHARACTER
0282 23                 INX     H       ;INCREMENT ADDRESS
0283 C37A02             JMP     SCRN
                ;
                ;
                ; THIS ROUTINE CONVERTS THE BINARY VALUE IN REG A INTO
                ; ASCII HEXADECIMAL DIGITS AND STORES THEM IN MEMORY.
                ;
0286 217410     BINH:   LXI     H,HCON  ;CONVERSION
0289 47                 MOV     B,A     ;SAVE VALUE
028A 1F                 RAR
028B 1F                 RAR
028C 1F                 RAR
028D 1F                 RAR
028E CD9902             CALL    BIN1
0291 77                 MOV     M,A
0292 23                 INX     H
0293 78                 MOV     A,B
0294 CD9902             CALL    BIN1    ;CONVERT TO ASCII
0297 77                 MOV     M,A
0298 C9                 RET
                ;
                ; THIS ROUTINE CONVERTS A VALUE TO HEXADECIMAL
                ;
0299 E60F       BIN1:   ANI     0FH     ;LOW 4 BITS
029B C630               ADI     48      ;CONVERT TO ASCII
029D FE3A               CPI     58      ;DIGIT 0-9
029F D8                 RC
02A0 C607               ADI     7       ;MODIFY FOR A-F
02A2 C9                 RET
                ;
                ;
                ; THIS ROUTINE CONVERTS THE BINARY VALUE IN REG A INTO
                ; ASCII DECIMAL DIGITS AND STORES THEM IN MEMORY
                ;
02A3 217410     BIND:   LXI     H,HCON  ;CONVERSION ADDRESS
02A6 0664               MVI     B,100
02A8 CDB402             CALL    BID1    ;CONVERT HUNDREDS DIGIT
02AB 060A               MVI     B,10
02AD CDB402             CALL    BID1    ;CONVERT TENS DIGIT
02B0 C630               ADI     '0'     ;GET UNITS DIGIT
02B2 77                 MOV     M,A     ;STORE IN MEMORY
02B3 C9                 RET
                ;
                ; THIS ROUTINE CONVERTS A VALUE TO DECIMAL
                ;
02B4 362F       BID1:   MVI     M,'0'-1 ;INITIALIZE DIGIT COUNT
02B6 34                 INR     M
02B7 90                 SUB     B       ;CHECK DIGIT
02B8 D2B602             JNC     BID1+2
02BB 80                 ADD     B       ;RESTORE VALUE
02BC 23                 INX     H
02BD C9                 RET
                ;
                ;
                ; LEGAL COMMAND TABLE
                ;
02BE 44554D50   CTAB:   DB      'DUMP'  ;DUMP COMMAND
02C2 0803               DW      DUMP    ;COMMAND ADDRESS
```

```
02C4 45584543          DB      'EXEC'  ;EXECUTE COMMAND
02C8 2101              DW      EXEC    ;COMMAND ADDRESS
02CA 454E5452          DB      'ENTR'  ;ENTER COMMAND
02CE 7604              DW      ENTR
02D0 46494C45          DB      'FILE'  ;FILE COMMAND
02D4 3E03              DW      FILE    ;COMMAND ADDRESS
02D6 4C495354          DB      'LIST'  ;LIST COMMAND
02DA D005              DW      LIST    ;COMMAND ADDRESS
02DC 44454C54          DB      'DELT'  ;DELETE COMMAND
02E0 E705              DW      DELL    ;COMMAND ADDRESS
02E2 4153534D          DB      'ASSM'  ;ASSEMBLE COMMAND
02E6 5E06              DW      ASSM    ;COMMAND ADDRESS
02E8 50414745          DB      'PAGE'  ;PAGE TRANSFER COMMAND
02EC 2203              DW      PAGE    ;COMMAND ADDRESS
02EE 43555354          DB      'CUST'  ;CUSTOMER COMMAND
02F2 0020              DW      2000H   ;COMMAND ADDRESS
02F4 4252454B          DB      'BREK'  ;BREAKPOINT COMMAND
02F8 D20C              DW      BREAK   ;COMMAND ADDRESS
02FA 50524F43          DB      'PROC'  ;;PROCEED COMMAND
02FE 8F0D              DW      PROC    ;COMMAND ADDRESS
                  ;
                  ;
                  ; THIS ROUTINE CHECKS IF ANY PARAMETERS WERE ENTERED
                  ; WITH THE COMMAND, IF NOT AN ERROR MESSAGE IS ISSUED
                  ;
0300 3A7E10       VCHK:   LDA     ABUF    ;FETCH PARAMETER BYTE
0303 B7                   ORA     A       ;SET FLAGS
0304 CA5A04               JZ      WHAT    ;NO PARAMETER
0307 C9                   RET
                  ;
                  ;
                  ; THIS ROUTINE DUMPS OUT THE CONTENTS OF MEMORY FROM
                  ; THE START TO FINAL ADDRESSES GIVEN IN THE COMMAND.
                  ;
0308 CD0003       DUMP:   CALL    VCHK    ;CHECK FOR PARAMETERS
030B CD0E01       DUMS:   CALL    CRLF    ;START NEW LINE
030E 2A8A10       DUMI:   LHLD    BBUF    ;FETCH MEMORY ADDRESS
0311 7E                   MOV     A,M
0312 CD4A02               CALL    HOTB    ;OUTPUT VALUE
0315 CD6302               CALL    ACHK    ;CHECK ADDRESS
0318 D8                   RC              ;RETURN IF FINISHED
0319 7D                   MOV     A,L     ;IS NEXT ADDRESS
031A E60F                 ANI     0FH     ;  DIVISIBLE BY 16?
031C C20E03               JNZ     DUMI
031F C30B03               JMP     DUMS
                  ;
                  ;
                  ; THIS ROUTINE WILL MOVE 256 BYTES FROM 1ST ADDRESS
                  ; GIVEN IN COMMAND TO 2ND ADDRESS IN COMMAND.
                  ;
0322 CD0003       PAGE:   CALL    VCHK    ;CHECK FOR PARAMETER
0325 3A8210               LDA     ABUF+4  ;FETCH 2ND PARAMETER
0328 B7                   ORA     A       ;DOES 2ND PARAMETER EXIST?
0329 CA5A04               JZ      WHAT
032C 2A8A10               LHLD    BBUF    ;FETCH MOVE TO ADDRESS
032F EB                   XCHG
0330 2A8C10               LHLD    BBUF+2  ;FETCH MOVE TO ADDRESS
0333 0600                 MVI     B,0     ;SET COUNTER
0335 1A           PAGI:   LDAX    D
0336 77                   MOV     M,A
0337 23                   INX     H
0338 13                   INX     D
0339 05                   DCR     B       ;DECREMENT COUNT
033A C23503               JNZ     PAGI
033D C9                   RET
                  ;
```

```
                ;
                ;
                ; THIS ROUTINE INITIALIZES THE BEGINNING OF FILE ADDRESS
                ; AND END OF FILE ADDRESS AS WELL AS THE FILE AREA
                ; WHEN THE FILE COMMAND IS USED
                ;
033E CD0E01     FILE:   CALL    CRLF
                ; CHECK FOR FILE PARAMETERS
0341 3A7610             LDA     FBUF
0344 B7                 ORA     A
0345 CA8903             JZ      FOUT    ;NO - GO LIST
0348 CD1804             CALL    FSEA    ;LOOK UP FILE
034B EB                 XCHG            ;PNTR IN DE
034C C26303             JNZ     TEST
                ; NO ENTRY
034F 3A7E10             LDA     ABUF    ;CHECK FOR PARAM
0352 B7                 ORA     A
0353 CA5D04             JZ      WHA1    ;NO?? - ERROR
                ; CHECK FOR ROOM IN DIRECTORY
0356 3A7D10             LDA     FEF
0359 B7                 ORA     A
035A C27803             JNZ     ROOM
035D 216B04             LXI     H,EMES1
0360 C36004             JMP     MESS
                ; ENTRY FOUND ARE THESE PARAMETERS
0363 3A7E10     TEST:   LDA     ABUF
0366 B7                 ORA     A
0367 CA8B03             JZ      SWAPS
036A 2A8A10             LHLD    BBUF
036D 7C                 MOV     A,H
036E B5                 ORA     L
036F CA8B03             JZ      SWAPS
0372 217004             LXI     H,EMES2 ;NO-NO CAN'T DO
0375 C36004             JMP     MESS    ;IT - DELETE FIRST
                ; MOVE FILE NAME TO BLOCK POINTED TO BY FREAD
0378 2A7B10     ROOM:   LHLD    FREAD
037B EB                 XCHG
037C 217610             LXI     H,FBUF  ;FILE NAME POINTER IN H,L
037F D5                 PUSH    D
0380 0E05               MVI     C,NMLEN ;NAME LENGTH COUNT
0382 7E         MOV23:  MOV     A,M
0383 12                 STAX    D
0384 13                 INX     D
0385 0D                 DCR     C       ;TEST COUNT
0386 23                 INX     H
0387 C28203             JNZ     MOV23
038A D1                 POP     D       ;RESTORE ENTRY POINTER
                ; MAKE FILE POINTED TO BY D,E CURRENT
038B 212410     SWAPS:  LXI     H,FILE0
038E 0E0D               MVI     C,FELEN ;ENTRY LENGTH
0390 1A         SWAP:   LDAX    D
0391 46                 MOV     B,M
0392 77                 MOV     M,A     ;EXCHANGE
0393 78                 MOV     A,B
0394 12                 STAX    D
0395 13                 INX     D
0396 23                 INX     H       ;BUMP POINTERS
0397 0D                 DCR     C       ;TEST COUNT
0398 C29003             JNZ     SWAP
                ; CHECK FOR 2ND PARAMETER
039B 3A7E10             LDA     ABUF
039E B7                 ORA     A
039F CAC303             JZ      FOOT    ;NO SECOND PARAMETER
                ; PROCESS 2ND PARAMETER
03A2 2A8A10             LHLD    BBUF    ;GET ADDRESS
03A5 222910             SHLD    BOFP    ;SET BEGIN
```

```
03A8 222B10             SHLD    EOFP     ;SET END
03AB 7D                 MOV     A,L      ;IS ADDRESS ZERO?
03AC B4                 ORA     H
03AD CAB203             JZ      FIL35    ;YES
03B0 3601       FIL30:  MVI     M,1      ;NON-ZERO - SET EOF
03B2 AF         FIL35:  XRA     A        ;AND MAX LINE #
03B3 322D10             STA     MAXL
03B6 C3C303             JMP     FOOT     ;OUTPUT PARAMETERS
03B9 3AC810     FOUT:   LDA     IBUF+4
03BC FE53               CPI     'S'      ;IS COMMAND FILES?
03BE 0E06               MVI     C,MAXFIL
03C0 CAC503             JZ      FOUL
03C3 0E01       FOOT:   MVI     C,1
                ; OUTPUT THE # OF ENTRIES IN C
03C5 212410     FOUL:   LXI     H,FILE0
03C8 79                 MOV     A,C
03C9 327D10     FINE:   STA     FOCNT    ;SAVE COUNT
03CC E5                 PUSH    H
03CD 110500             LXI     D,NMLEN
03D0 19                 DAD     D
03D1 7E                 MOV     A,M
03D2 B7                 ORA     A
03D3 C2E303             JNZ     FOOD     ;NON ZERO, OK TO OUTPUT
03D6 23                 INX     H
03D7 86                 ADD     M
03D8 23                 INX     H
03D9 C2E303             JNZ     FOOD
03DC 33                 INX     SP
03DD 33                 INX     SP
03DE 23                 INX     H
03DF 23                 INX     H
03E0 C3F803             JMP     FEET
                ; HAVE AN ENTRY TO OUTPUT
03E3 E1         FOOD:   POP     H        ;PTR
03E4 0E05               MVI     C,NMLEN
03E6 46         FAST:   MOV     B,M      ;LOAD CHARACTER TO B
03E7 CD0301             CALL    OUT8
03EA 0D                 DCR     C
03EB 23                 INX     H
03EC C2E603             JNZ     FAST     ;DO THE REST
                ; NOW OUTPUT BEGIN-END PTRS
03EF CD0404             CALL    FOOL     ;OUTPUT BEGIN
03F2 CD0404             CALL    FOOL     ;OUTPUT END
03F5 CD0E01             CALL    CRLF     ;AND C/R
                ; TEST COUNT, H,L POINTS PAST EOFP
03F8 110400     FEET:   LXI     D,FELEN-NMLEN-4
03FB 19                 DAD     D        ;MOVE TO NEXT ENTRY
03FC 3A7D10             LDA     FOCNT
03FF 3D                 DCR     A        ;TEST COUNT
0400 C2C903             JNZ     FINE     ;MORE TO DO
0403 C9                 RET              ;DONE!
                ; OUTPUT NUMBER POINTED TO BY H,L
                ; ON RET, H,L POINT 2 WORDS LATER
0404 CD5D02     FOOL:   CALL    BLK1     ;SPACE
0407 23                 INX     H
0408 7E                 MOV     A,M
0409 2B                 DCX     H
040A E5                 PUSH    H
040B CD3A02             CALL    HOUT     ;OUTPUT
040E E1                 POP     H
040F 7E                 MOV     A,M
0410 23                 INX     H
0411 23                 INX     H
0412 E5                 PUSH    H
0413 CD4A02             CALL    HOTB     ;OUTPUT
0416 E1                 POP     H        ;RESTORE H,L
```

```
0417 C9                  RET
                ; SEARCH THE FILE DIRECTORY FOR THE FILE
                ; WHOSE NAME IS IN FBUF.
                ; RETURN IF FOUND, ZERO IS OFF, H,L POINT TO
                ; ENTRY WHILE SEARCHING, ON ENTRY FOUND WITH ADDR
                ; ZERO, SET FEF TO >0 AND FREAD TO THE ADDR OF ENTRY
                ;
0418 AF         FSEA:    XRA      A
0419 327D10              STA      FEF      ;CLAIM NO FREE ENTRIES
041C 0606                MVI      B,MAXFIL        ;COUNT OF ENTRIES
041E 112410              LXI      D,FILE0 ;TABLE ADDRESS
0421 217610     FSE10:   LXI      H,FBUF
0424 0E05                MVI      C,NMLEN
0426 CD5301              CALL     SEAR     ;TEST STRINGS
0429 F5                  PUSH     PSW      ;SAVE FLAG
042A D5                  PUSH     D
042B 1A                  LDAX     D        ;GET BOFP
042C B7                  ORA      A        ;EMPTY ENTRY?
042D C24E04              JNZ      FSE20
0430 13                  INX      D        ;STORE OTHER WORD
0431 1A                  LDAX     D
0432 B7                  ORA      A
0433 C24E04              JNZ      FSE20    ;NOPE-GO TEST FOR MATCH
0436 EB                  XCHG
0437 11FAFF              LXI      D,-NMLEN-1
043A 19                  DAD      D        ;MOV TO BEGINNING
043B 227B10              SHLD     FREAD    ;SAVE ADDR
043E 7A                  MOV      A,D
043F 327D10              STA      FEF      ;SET FREE ENTRY FOUND
0442 E1                  POP      H        ;RESTOR INTERIM PTR
0443 F1                  POP      PSW      ;UNJUNK STACK
                ; MOVE TO NEXT ENTRY
0444 110800     FSE15:   LXI      D,FELEN-NMLEN
0447 19                  DAD      D
0448 EB                  XCHG              ;NEXT ENTRY ADDR IN DE
0449 05                  DCR      B        ;TEST COUNT
044A C8                  RZ                ;DONE--NOPE
044B C32104              JMP      FSE10    ;TRY NEXT
                ;ENTRY WASN'T FREE, TEST FOR MATCH
044E E1         FSE20:   POP      H
044F F1                  POP      PSW
0450 C24404              JNZ      FSE15    ;IF ZERO CLEAR, NO MATCH
                ;ENTRY FOUND
0453 11FBFF              LXI      D,-NMLEN        ;BACKUP
0456 19                  DAD      D        ;H,L POINTS TO ENTRY
0457 7A                  MOV      A,D
0458 B7                  ORA      A        ;CLEAR ZERO
0459 C9                  RET               ;THAT'S ALL
                ;
                ;
                ; OUTPUT ERROR MESSAGE FOR ILLEGAL COMMAND
                ;
045A CD0E01     WHAT:    CALL     CRLF     ;OUT CRLF
045D 216604     WHA1:    LXI      H,EMES   ;MESSAGE ADDRESS
0460 CD7A02     MESS:    CALL     SCRN
0463 C36700              JMP      EOR
                ;
0466 57484154 EMES:      DB       'WHAT'
046A 0D                  DB       13       ;CARRIAGE RETURN
046B 46554C4C EMES1:     DB       'FULL',13
046F 0D
0470 4E4F204E EMES2:     DB       'NO NO',13
0474 4F0D
                ;
                ;
                : CALL ROUTINE TO ENTER DATA INTO MEMORY
                ; AND CHECK FOR ERROR ON RETURN
                ;
                ; THIS ROUTINE IS USED TO ENTER DATA VALUES INTO MEMORY.
                ; EACH VALUE IS ONE BYTE AND IS WRITTEN IN HEXADECIMAL
                ; VALUES GREATER THAT 255 WILL CAUSE CARRY TO BE SET
                ; AND RETURN TO BE MADE TO CALLING PROGRAM
                ;
0476 CD0003     ENTR:   CALL    VCHK    ;CHECK FOR PARAMETERS
0479 CD8304             CALL    ENTS
047C DA5A04             JC      WHAT
047F CD0E01             CALL    CRLF
0482 C9                 RET
                ;
                ;
002F            EEND    EQU     '/'     ;TERMINATION CHAR
0483 CD0E01     ENTS:   CALL    CRLF
0486 CD8000             CALL    READ    ;READ INPUT DATA
0489 21C710             LXI     H,IBUF  ;SET LINE POINTER
048C 229610             SHLD    PNTR    ;SAVE POINTER
048F CD6601     ENT1:   CALL    ZBUF    ;CLEAR BUFFER
0492 CD0D09             CALL    SBLK    ;CAN TO FIRST VALUE
0495 DA8304             JC      ENTS    ;JUMP IF CR FOUND
0498 FE2F               CPI     EEND
049A C8                 RZ              ;RETURN CARRY IS ZERO
049B CD7508             CALL    ALPS    ;PLACE VALUE IN BUFFER
049E 78                 MOV     A,B     ;GET DIGIT COUNT
049F FE03               CPI     3       ;CHECK NUR OF DIGITS
04A1 3F                 CMC
04A2 D8                 RC              ;RETURN IF MORE THAN    2 DIGITS
04A3 017E10             LXI     B,ABUF  ;CONVERSION ADDRESS
04A6 CD1B02             CALL    AHEX    ;CONVERT VALUE
04A9 D8                 RC              ;ERROR IN HEX CHARACTER
04AA 7D                 MOV     A,L
04AB 2A8A10             LHLD    BBUF    ;FETCH MEMORY ADDRESS
04AE 77                 MOV     M,A     ;PUT IN MEMORY
04AF CD7502             CALL    ACH1    ;INCREMENT MEMORY LOCATION
04B2 C38F04             JMP     ENT1
                ;
                ;
                ;
                ; THIS ROUTINE IS USED TO ENTER LINES INTO THE FILE
                ; AREA. THE LINE NUMBER IS FIRST CHECKED TO SEE IF IT IS
                ; A VALID NUMBER (0000-9999). NEXT IT IS CHECKED TO SEE
                ; IF IT IS GREATER THAN THE MAXIMUM CURRENT LINE NUMBER.
                ; IF IT IS, THE NEXT LINE IS INSERTED AT THE END OF THE
                ; CURRENT FILE AND THE MAXIMUM LINE NUMBER IS UPDATED AS
                ; WELL AS THE END OF FILE POSITION. LINE NUMBERS THAT
                ; ALREADY EXIST ARE INSERTED INTO THE FILE AREA AT THE
                ; APPROPRIATE PLACE AND ANY EXTRA CHARACTERS IN THE OLD
                ; LINE ARE DELETED.
                ;
04B5 3A2410     LINE:   LDA     FILE0   ;IS A FILE DEFINED?...
04B8 B7                 ORA     A
04B9 CA5A04             JZ      WHAT    ;ABORT IF NOT
04BC 0E04               MVI     C,4     ;NO OF DIGITS TO CHECK
04BE 21C510             LXI     H,IBUF-1        ;INITIALIZE ADDRESS
04C1 23         LICK:   INX     H
04C2 7E                 MOV     A,M     ;FETCH LINE DIGIT
04C3 FE30               CPI     '0'     ;CHECK FOR VALID NUMBER
04C5 DA5A04             JC      WHAT
04C8 FE3A               CPI     '9'+1
04CA D25A04             JNC     WHAT
04CD 0D                 DCR     C
04CE C2C104             JNZ     LICK
04D1 227410             SHLD    ADDS    ;FIND ADDRESS
04D4 113010             LXI     D,MAXL+3        ;GET ADDRESS
```

```
04D7 CDA205            CALL    COMO
04DA D2FA04            JNC     INSR
              ; GET HERE IF NEW LINE IS GREATER THAN MAXIMUM LINE #
04DD 23                INX     H
04DE CD9205            CALL    LODM     ;GET NEW LINE NUMBER
04E1 213010            LXI     H,MAXL+3
04E4 CD9A05            CALL    STOM     ;MAKE IT MAXIMUM LINE NUMBER
04E7 11C610            LXI     D,IBUF-1
04EA 2A2B10            LHLD    EOFP     ;END OF FILE POSITION
04ED 0E01              MVI     C,1
04EF CD8005            CALL    LMOV     ;PLACE LINE IN FILE
04F2 3601     SEOF:    MVI     M,1      ;END OF FILE INDICATOR
04F4 222B10            SHLD    EOFP     ;END OF FILE ADDRESS
04F7 C36700            JMP     EOR
              ; GET HERE IF NEW LINE MUST BE INSERTED INTO ALREADY
              ; EXISTING FILE AREA
04FA CD5205   INSR:    CALL    FINI     ;FIND LINE IN FILE
04FD 0E02              MVI     C,2
04FF CA0305            JZ      EQUL
0502 0D                DCR     C        ;NEW LN NOT EQUAL TO SOME OLD LN
0503 46       EQUL:    MOV     B,M
0504 2B                DCX     H
0505 3602              MVI     M,2      ;MOVE LINE INDICATOR
0507 227210            SHLD    INSP     ;INSERT LINE POSITION
050A 3AC610            LDA     IBUF-1   ;NEW LN COUNT
050D 0D                DCR     C
050E CA1805            JZ      LT       ;NEW LN NOT = OLD LN
0511 90                SUB     B        ;COUNT DIFFERENCE
0512 CA3B05            JZ      ZERO     ;LINE LENGTHS EQUAL
0515 DA2B05            JC      GT
              ; GET HERE IF # OF CHARS IN OLD LINE > # OF CHARS IN
              ; NEW LINE OR NEW LINE # WAS NOT EQUAL TO SOME OLD
              ; LINE #
0518 2A2B10   LT:      LHLD    EOFP     ;END OF FILE ADDRESS
051B 54                MOV     D,H
051C 5D                MOV     E,L
051D CD7B05            CALL    ADR      ;MOVE TO ADDRESS
0520 222B10            SHLD    EOFP     ;NEW END OF FILE ADDRESS
0523 0E02              MVI     C,2
0525 CD8905            CALL    RMOV     ;OPEN UP FILE AREA
0528 C33B05            JMP     ZERO
              ; GET HERE IF # OF CHARS IN OLD LINE < # OF CHARS IN
              ; NEW LINE.
052B 2F       GT:      CMA
052C 3C                INR     A        ;COUNT DIFFERENCE
052D 54                MOV     D,H
052E 5D                MOV     E,L
052F CD7B05            CALL    ADR
0532 EB                XCHG
0533 CD8005            CALL    LMOV     ;DELETE EXCESS CHAR IN FILE
0536 3601              MVI     M,1      ;E-O-F INDICATOR
0538 222B10            SHLD    EOFP     ;E-O-F ADDRESS
              ; GET HERE TO INSERT CURRENT LINE INTO FILE AREA
053B 2A7210   ZERO:    LHLD    INSP     ;INSERT ADDRESS
053E 360D              MVI     M,ASCR
0540 23                INX     H
0541 11C610            LXI     D,IBUF-1          ;NEW LINE ADDRESS
0544 0E01              MVI     C,1      ;CHECK VALUE
0546 CD8005            CALL    LMOV     ;PLACE LINE IN FILE
0549 C36700            JMP     EOR
              ;
              ;
              ; THIS ROUTINE IS USED TO FIND A LN IN THE FILE AREA
              ; WHICH IS GREATER THAN OR EQUAL TO THE CURRENT LINE #
054C 218110   FIND:    LXI     H,ABUF+3          ;BUFFER ADDRESS
054F 227410            SHLD    ADDS     ;SAVE ADDRESS
```

```
0552 2A2910     FIN1:   LHLD    BOFP     ;BEGIN FILE ADDRESS
0555 7C                 MOV     A,H      ;RETURN TO MONITOR IF
0556 B5                 ORA     L        ;  FILE IS EMPTY...
0557 CA6700             JZ      EOR
055A CD7405     FI1:    CALL    EO1      ;CHECK FOR END OF FILE
055D EB                 XCHG
055E 2A7410             LHLD    ADDS     ;FETCH FIND ADDRESS
0561 EB                 XCHG
0562 3E04               MVI     A,4
0564 CD7B05             CALL    ADR      ;BUMP LINE ADDRESS
0567 CDA205             CALL    COM0     ;COMPARE LINE NUMBERS
056A D8                 RC
056B C8                 RZ
056C 7E         FI2:    MOV     A,M
056D CD7B05             CALL    ADR      ;NEXT LINE ADDRESS
0570 C35A05             JMP     FI1
                ;
                ;
                ; WHEN SEARCHING THROUGH THE FILE AREA, THIS ROUTINE
                ; CHECKS TO SEE IF THE CURRENT ADDRESS IS THE END OF
                ; FILE
                ;
0573 23         EOF:    INX     H
0574 3E01       EO1:    MVI     A,1      ;E-O-F INDICATOR
0576 BE                 CMP     M
0577 C0                 RNZ
0578 C36700             JMP     EOR
                ;
                ;
                ; THIS ROUTINE IS USED TO ADD A VALUE TO AN ADDRESS
                ; CONTAINED IN REGISTER H,L
                ;
057B 85         ADR:    ADD     L
057C 6F                 MOV     L,A
057D D0                 RNC
057E 24                 INR     H
057F C9                 RET
                ;
                ;
                ; THIS ROUTINE WILL MOVE CHARACTER STRINGS FROM ONE
                ; LOCATION OF MEMORY TO ANOTHER
                ; CHARACTERS ARE MOVED FROM LOCATION ADDRESSED BY D,E
                ; TO LOCATION ADDRESSED BY H,L.  ADDITIONAL CHARACTERS
                ; ARE MOVED BY BUMPING POINTERS UNTIL THE CHARACTER IN
                ; REG C IS FETCHED.
                ;
0580 1A         LMOV:   LDAX    D        ;FETCH CHARACTER
0581 13                 INX     D        ;INCREMENT FETCH ADDRESS
0582 B9                 CMP     C        ;TERMINATION CHARACTER
0583 C8                 RZ
0584 77                 MOV     M,A      ;STORE CHARACTER
0585 23                 INX     H        ;INCREMENT STORE ADDRESS
0586 C38005             JMP     LMOV
                ;
                ;
                ; THIS ROUTINE IS SIMILAR TO ABOVE EXCEPT THAT THE
                ; CHARACTER ADDRESS IS DECREMENTED AFTER EACH FETCH
                ; AND STORE
                ;
0589 1A         RMOV:   LDAX    D        ;FETCH CHARACTER
058A 1B                 DCX     D        ;DECREMENT FETCH ADDRESS
058B B9                 CMP     C        ;TERMINATION CHARACTER
058C C8                 RZ
058D 77                 MOV     M,A      ;STORE CHARACTER
058E 2B                 DCX     H        ;DECREMENT STORE ADDRESS
058F C38905             JMP     RMOV
```

```
                    ;
                    ;
                    ; THIS ROUTINE IS USED TO LOAD FOUR CHARACTERS FROM
                    ; MEMORY INTO REGISTERS
                    ;
0592 46             LODM:   MOV     B,M     ;FETCH CHARACTER
0593 23                     INX     H
0594 4E                     MOV     C,M     ;FETCH CHARACTER
0595 23                     INX     H
0596 56                     MOV     D,M     ;FETCH CHARACTER
0597 23                     INX     H
0598 5E                     MOV     E,M     ;FETCH CHARACTER
0599 C9                     RET
                    ;
                    ;
                    ; THIS ROUTINE STORES FOUR CHARACTERS FROM THE REGISTERS
                    ; INTO MEMORY
                    ;
059A 73             STOM:   MOV     M,E     ;STORE CHARACTER
059B 2B                     DCX     H
059C 72                     MOV     M,D     ;STORE CHARACTER
059D 2B                     DCX     H
059E 71                     MOV     M,C     ;STORE CHARACTER
059F 2B                     DCX     H
05A0 70                     MOV     M,B     ;STORE CHARACTER
05A1 C9                     RET
                    ;
                    ;
                    ; THIS ROUTINE IS USED TO COMPARE TWO CHARACTER STRINGS
                    ; OF LENGTH 4, ON RETURN ZERO FLAG SET MEANS BOTH
                    ; STRINGS ARE EQUAL.  CARRY FLAG =0 MEANS STRING ADDRESS
                    ; BY D,E WAS GREATER THAN OR EQUAL TO CHARACTER STRING
                    ; ADDRESSED BY H,L
                    ;
05A2 0601           COM0:   MVI     B,1     ;EQUAL COUNTER
05A4 0E04                   MVI     C,4     ;STRING LENGTH
05A6 B7                     ORA     A       ;CLEAR CARRY
05A7 1A             CO1:    LDAX    D       ;FETCH CHARACTER
05A8 9E                     SBB     M       ;COMPARE CHARACTERS
05A9 CAAD05                 JZ      CO2
05AC 04                     INR     B       ;INCREMENT EQUAL COUNTER
05AD 1B             CO2:    DCX     D
05AE 2B                     DCX     H
05AF 0D                     DCR     C
05B0 C2A705                 JNZ     CO1
05B3 05                     DCR     B
05B4 C9                     RET
                    ;
                    ; THIS ROUTINE IS SIMILAR TO THE ABOVE ROUTINE EXCEPT ON
                    ; RETURN CARRY FLAG = 0 MEANS THAT CHARACTER STRING
                    ; ADDRESSED BY D,E IS ONLY > STRING ADDRESSED BY H,L.
                    ;
05B5 0E04           COM1:   MVI     C,4     ;STRING LENGTH
05B7 1A                     LDAX    D       ;TCH CHARACTER
05B8 D601                   SUI     1
05BA C3A805                 JMP     CO1+1
                    ;
                    ;
                    ; THIS ROUTINE WILL TAKE ASCII CHARACTERS AND ADD ANY
                    ; NECESSARY ASCII ZEROES SO THE RESULT IS A 4 CHARACTER
                    ; ASCII VALUE
                    ;
05BD CD9205         NORM:   CALL    LODM    ;LOAD CHARACTERS
05C0 AF                     XRA     A       ;FETCH A ZERO
05C1 B8                     CMP     B
05C2 C8                     RZ
```

```
05C3 BB         NORI:   CMP     E
05C4 C49A05             CNZ     STOM     ;STORE VALUES
05C7 C0                 RNZ
05C8 5A                 MOV     E,D      ;NORMALIZE VALUE
05C9 51                 MOV     D,C
05CA 48                 MOV     C,B
05CB 0630               MVI     B,'0'
05CD C3C305             JMP     NORI
                ;
                ;
                ; THIS ROUTINE IS USED TO LIST THE CONTENTS OF THE FILE
                ; AREA STARTING AT THE LINE NUMBER GIVEN IN THE COMMAND
                ;
05D0 CD0E01     LIST:   CALL    CRLF
05D3 CD4C05             CALL    FIND     ;FIND STARTING LN
05D6 23         LIST0:  INX     H        ;OUTPUT LINE...
05D7 CD7A02             CALL    SCRN
05DA CD0E01             CALL    CRLF
05DD CD7305             CALL    EOF      ;CHECK FOR END OF FILE
05E0 CDE900             CALL    INK      ;CHECK FOR ↑X
05E3 C2D605             JNZ     LIST0    ;LOOP IF NO ↑X
05E6 C9                 RET
                ;
                ;
                ; THIS ROUTINE IS USED TO DELETE LINES FROM THE
                ; FILE AREA.  THE REMAINING FILE AREA IS THEN MOVED IN
                ; MEMORY SO THAT THERE IS NO EXCESS SPACE.
                ;
05E7 CD0003     DELL:   CALL    VCHK     ;CHECK FOR PARAMETER
05EA CD4C05             CALL    FIND     ;FIND LINE IN FILE AREA
05ED 227210             SHLD    DELP     ;SAVE DELETE POSITION
05F0 218510             LXI     H,ABUF+7
05F3 7E                 MOV     A,M      ;CHECK FOR 2ND PARAMETER
05F4 B7                 ORA     A        ;SET FLAGS
05F5 C2FB05             JNZ     DEL1
05F8 218110             LXI     H,ABUF+3         ;USE FIRST PARAMETER
05FB 227410     DEL1:   SHLD    ADDS     ;SAVE FIND ADDRESS
05FE EB                 XCHG
05FF 213010             LXI     H,MAXL+3
0602 CDA205             CALL    COM0     ;COMPARE LINE NUMBERS
0605 2A7210             LHLD    DELP     ;LOAD DELETE POSITION
0608 DA4906             JC      NOVR
                ; GET HERE IF DELETION INVOLVES END OF FILE
060B 222B10             SHLD    EOFP     ;CHANGE E-O-F POSITION
060E 3601               MVI     M,1      ;SET E-O-F INDICATOR
0610 EB                 XCHG
0611 2A2910             LHLD    BOFP     ;GET BEGIN FILE ADDRESS
0614 EB                 XCHG
0615 060D               MVI     B,13     ;SET SCAN SWITCH
0617 2B                 DCX     H        ;CHECK FOR BOF
0618 7D         DEL2:   MOV     A,L
0619 93                 SUB     E
061A 7C                 MOV     A,H
061B 9A                 SBB     D
061C 3E0D               MVI     A,ASCR   ;LOOK FOR CR
061E DA4006             JC      DEL4     ;DECREMENTED PAST BOF
0621 05                 DCR     B
0622 2B                 DCX     H
0623 BE                 CMP     M        ;FIND NEW MAX LN
0624 C21806             JNZ     DEL2
0627 2B                 DCX     H
0628 7D                 MOV     A,L
0629 93                 SUB     E
062A 7C                 MOV     A,H
062B 9A                 SBB     D
062C DA4106             JC      DEL5
```

```
062F BE                 CMP     M        ;END OF PREVIOUS LINE
0630 23                 INX     H
0631 23                 INX     H
0632 CA3606             JZ      DEL3
0635 23                 INX     H
0636 CD9205    DEL3:    CALL    LODM     ;LOAD NEW MAX LN
0639 213010             LXI     H,MAXL+3         ;SET ADDRESS
063C CD9A05             CALL    STOM     ;STORE NEW MAX LN
063F C9                 RET
0640 B8        DEL4:    CMP     B        ;CHECK SWITCH
0641 EB        DEL5:    XCHG
0642 C23506             JNZ     DEL3-1
0645 322D10             STA     MAXL     ;MAKE MAX LN A SMALL NUMBER
0648 C9                 RET
               ; GET HERE IF DELETION IS IN MIDDLE OF FILE AREA
0649 CD5A05    NOVR:    CALL    FI1      ;FIND END OF DELETE AREA
064C CC6C05             CZ      FI2      ;NEXT LINE IF THIS LN EQUAL
064F EB        NOV1:    XCHG
0650 2A7210             LHLD    DELP     ;CHAR MOVE TO POSITION
0653 0E01               MVI     C,1      ;MOVE TERMINATOR
0655 CD8005             CALL    LMOV     ;COMPACT FILE AREA
0658 222B10             SHLD    EOFP     ;SET EOF POSITION
065B 3601               MVI     M,1      ;SET EOF INDICATOR
065D C9                 RET
               ;
               ; STARTING HERE IS THE SELF ASSEMBLER PROGRAM
               ; THIS PROGRAM ASSEMBLES PROGRAMS WHICH ARE
               ; IN THE FILE AREA
               ;
065E CD0003    ASSM:    CALL    VCHK     ;CHECK FOR PARAMETER
0661 3A8210             LDA     ABUF+4   ;GET 2ND PARAMETER
0664 B7                 ORA     A        ;CHECK FOR PARAMETERS
0665 C26E06             JNZ     ASM4
0668 2A8A10             LHLD    BBUF     ;FETCH 1ST PARAMETER
066B 228C10             SHLD    BBUF+2   ;STORE INTO 2ND PARAMETER
066E 3ACB10    ASM4:    LDA     IBUF+4   ;FETCH INPUT CHARACTER
0671 D645               SUI     'E'      ;RESET A IF ERRORS ONLY
0673 328E10             STA     AERR     ;SAVE ERROR FLAG
0676 AF                 XRA     A        ;GET A ZERO
0677 329810             STA     NOLA     ;INITIALIZE LABEL COUNT
067A 329410    ASM3:    STA     PASI     ;SET PASS INDICATOR
067D CD0E01             CALL    CRLF     ;INDICATE START OF PASS
0680 2A8A10             LHLD    BBUF     ;FETCH ORIGIN
0683 229210             SHLD    ASPC     ;INITIALIZE PC
0686 2A2910             LHLD    BOFP     ;GET START OF FILE
0689 227210             SHLD    APNT
068C 2A7210    ASM1:    LHLD    APNT     ;FETCH LINE POINTER
068F 31B210             LXI     SP,AREA+18
0692 7E                 MOV     A,M      ;FETCH CHARACTER
0693 FE01               CPI     1        ;END OF FILE?
0695 CA0109             JZ      EASS     ;JUMP IF END OF FILE
0698 EB                 XCHG
0699 13                 INX     D        ;INCREMENT ADDRESS
069A 21B210             LXI     H,OBUF   ;BLANK START ADDRESS
069D 3EC2               MVI     A,IBUF-5 AND 0FFH ;BLANK END ADDRESS
069F CDE100             CALL    CLER     ;BLANK OUT BUFFER
06A2 0E0D               MVI     C,ASCR   ;STOP CHARACTER
06A4 CD8005             CALL    LMOV     ;MOVE LINE INTO BUFFER
06A7 71                 MOV     M,C      ;PLACE CR IN BUFFER
06A8 EB                 XCHG
06A9 227210             SHLD    APNT     ;SAVE ADDRESS
06AC 3A9410             LDA     PASI     ;FETCH PASS INDICATOR
06AF B7                 ORA     A        ;SET FLAGW
06B0 C2B906             JNZ     ASM2     ;JUMP IF PASS 2
06B3 CDCC06             CALL    PAS1
06B6 C38C06             JMP     ASM1
```

```
06B9 CD9307   ASM2:   CALL    PAS2
06BC 21B210           LXI     H,OBUF    ;OUTPUT BUFFER ADDRESS
06BF CDC506           CALL    AOUT      ;OUTPUT LINE
06C2 C38C06           JMP     ASM1
              ;
              ; THIS ROUTINE IS USED TO OUTPUT THE LISTING FOR
              ; AN ASSEMBLY.  IT CHECKS THE ERROR SWITCH TO SEE IF
              ; ALL LINES ARE TO BE PRINTED OR JUST THOSE WITH
              ; ERRORS.
              ;
06C5 3A8E10   AOUT:   LDA     AERR      ;FETCH ERROR SWITCH
06C8 B7               ORA     A         ;SET FLAGS
06C9 C2D206           JNZ     AOU1      ;OUTPUT ALL LINES
06CC 3AB210   AOU2:   LDA     OBUF      ;FETCH ERROR INDICATOR
06CF FE20             CPI     ' '       ;CHECK FOR AN ERROR
06D1 C8               RZ                ;RETURN IF NO ERROR
06D2 21B210   AOU1:   LXI     H,OBUF    ;OUTPUT BUFFER ADDRESS
06D5 CD7A02           CALL    SCRN      ;OUTPUT LINE...
06D8 CD0E01           CALL    CRLF
06DB C9               RET
              ;
              ; PASS1 OF ASSEMBLER. USED TO FORM SYMBOL TABLE
              ;
06DC CD6601   PAS1:   CALL    ZBUF      ;CLEAR BUFFER
06DF 329410           STA     PASI      ;SET FOR PASS1
06E2 21C710           LXI     H,IBUF    ;INITIALIZE LINE POINTER
06E5 229610           SHLD    PNTR
06E8 7E               MOV     A,M       ;FETCH CHARACTER
06E9 FE20             CPI     ' '       ;CHECK FOR A BLANK
06EB CA1E07           JZ      OPC       ;JUMP IF NO LABEL
06EE FE2A             CPI     '*'       ;CHECK FOR COMMENT
06F0 C8               RZ                ;RETURN IF COMMENT
              ;
              ; PROCESS LABEL
              ;
06F1 CD2008           CALL    SLAB      ;GET AND CHECK LABEL
06F4 DADF0A           JC      OP5       ;ERROR IN LABEL
06F7 CAC70C           JZ      ERRD      ;DUPLICATE LABEL
06FA CD3507           CALL    LCHK      ;CHECK CHARACTER AFTER LABEL
06FD C2DF0A           JNZ     OP5       ;ERROR IF NO BLANK
0700 0E05             MVI     C,LLAB    ;LENGTH OF LABELS
0702 217E10           LXI     H,ABUF    ;SET BUFFER ADDRESS
0705 7E       MLAB:   MOV     A,M       ;FETCH NEXT CHARACTER
0706 12               STAX    D         ;STORE IN SYMBOL TABLE
0707 13               INX     D
0708 23               INX     H
0709 0D               DCR     C
070A C20507           JNZ     MLAB
070D EB               XCHG
070E 229010           SHLD    TABA      ;SAVE TABLE ADDRESS FOR EQU
0711 3A9310           LDA     ASPC+1    ;FETCH PC (HIGH)
0714 77               MOV     M,A
0715 23               INX     H
0716 3A9210           LDA     ASPC      ;FETCH PC (LOW)
0719 77               MOV     M,A       ;STORE IN TABLE
071A 219810           LXI     H,NOLA
071D 34               INR     M         ;INCREMENT NUMBER OF LABELS
              ;
              ; PROCESS OPCODE
              ;
071E CD6601   OPC:    CALL    ZBUF      ;ZERO WORKING BUFFER
0721 CD0D09           CALL    SBLK      ;SCAN TO OPCODE
0724 DA0608           JC      OERR      ;FOUND CARRIAGE RETURN
0727 CD7508           CALL    ALPS      ;PLACE OPCODE IN BUFFER
072A FE20             CPI     ' '       ;CHECK FOR BLANK AFTER OPCODE
072C DA650A           JC      OPCD      ;CR AFTER OPCODE
```

```
072F C2060B           JNZ     OERR     ;ERROR IF NO BLANK
0732 C3650A           JMP     OPCD     ;CHECK OPCODE
              ;
              ; THIS ROUTINE CHECKS THE CHARACTER AFTER A LABEL
              ; FOR A BLANK OR A COLON.
              ;
0735 2A9610   LCHK:   LHLD    PNTR
0738 7E               MOV     A,M      ;GET CHARACTER AFTER LABEL
0739 FE20             CPI     ' '      ;CHECK FOR A BLANK
073B C8               RZ               ;RETURN IF A BLANK
073C FE3A             CPI     ':'      ;CHECK FOR A COLON
073E C0               RNZ
073F 23               INX     H
0740 229610           SHLD    PNTR     ;SAVE POINTER
0743 C9               RET
              ;
              ; PROCESS ANY PSEUDO OPS THAT NEED TO BE IN PASS 1
              ;
0744 CD0D09   PSU1:   CALL    SBLK     ;SCAN TO OPERAND
0747 1A               LDAX    D        ;FETCH VALUE
0748 B7               ORA     A        ;SET FLAGS
0749 CA6007           JZ      ORG1     ;ORG OPCODE
074C FA9007           JM      DAT1     ;DATA STATEMENT
074F E27507           JPO     EQU1     ;EQU OPCODE
0752 FE05             CPI     5
0754 DA8807           JC      RES1     ;RES OPCODE
0757 C20109           JNZ     EASS     ;JUMP IF END
              ; DO DW PSEUDO/OP
075A 0E02     AC01:   MVI     C,2      ;2 BYTE INSTRUCTION
075C AF               XRA     A        ;GET A ZERO
075D C3F50A           JMP     OCN1     ;ADD VALUE TO PROGRAM CNTR
              ; DO ORG PSEUDO-OP
0760 CD970B   ORG1:   CALL    ASCN     ;GET OPERAND
0763 3A8210           LDA     OBUF     ;FETCH ERROR INDICATOR
0766 FE20             CPI     ' '      ;CHECK FOR AN ERROR
0768 C0               RNZ
0769 229210           SHLD    ASPC     ;STORE NEW ORIGIN
076C 3AC710           LDA     IBUF     ;GET FIRST CHARACTER
076F FE20             CPI     ' '      ;CHECK FOR LABEL
0771 C8               RZ               ;NO LABEL
0772 C38007           JMP     EQUS     ;CHANGE LABEL VALUE
              ; DO EQU PSEUDO-OP
0775 CD970B   EQU1:   CALL    ASCN     ;GET OPERAND
0778 3AC710           LDA     IBUF     ;FETCH 1ST CHARACTER
077B FE20             CPI     ' '      ;CHECK FOR LABEL
077D CA9F0C           JZ      ERRM     ;MISSING LABEL
0780 EB       EQUS:   XCHG
0781 2A9010           LHLD    TABA     ;SYMBOL TABLE ADDRESS
0784 72               MOV     M,D      ;STORE LABEL VALUE
0785 23               INX     H
0786 73               MOV     M,E
0787 C9               RET
              ; DO DS PSEUDO-OP
0788 CD970B   RES1:   CALL    ASCN     ;GET OPERAND
078B 44               MOV     B,H
078C 4D               MOV     C,L
078D C3ED07           JMP     RES21    ;ADD VALUE TO PROGRAM COUNTER
              ;
              ; DO DB PSEUDO-OP
              ;
0790 C3F407   DAT1:   JMP     DAT2A
              ;
              ; PERFORM PASS 2 OF THE ASSEMBLER
              ;
0793 218410   PAS2:   LXI     H,OBUF+2 ;SET OUTPUT BUFFER ADDRESS
0796 3A9310           LDA     ASPC+1   ;FETCH PC(HIGH)
```

```
0799 CD8902            CALL    BINH+3  ;CONVERT FOR OUTPUT
079C 23                INX     H
079D 3A9210            LDA     ASPC    ;FETCH PC(LOW)
07A0 CD8902            CALL    BINH+3  ;CONVERT FOR OUTPUT
07A3 23                INX     H
07A4 229E10            SHLD    OIND    ;SAVE OUTPUT ADDRESS
07A7 CD6601            CALL    ZBUF    ;CLEAR BUFFER
07AA 21C710            LXI     H,IBUF  ;INITIALIZE LINE POINTER
07AD 229610    PABL:   SHLD    PNTR    ;SAVE POINTER
07B0 7E                MOV     A,M     ;FETCH FIRST CHARACTER
07B1 FE20              CPI     ' '     ;CHECK FOR LABEL
07B3 CA1E07            JZ      OPC     ;GET OPCODE
07B6 FE2A              CPI     '*'     ;CHECK FOR COMMENT
07B8 C8                RZ              ;RETURN IF COMMENT
07B9 CD2008            CALL    SLAB    ;SCAN OFF LABEL
07BC DAC20C            JC      ERRL    ;ERROR IN LABEL
07BF CD3507            CALL    LCHK    ;CHECK FOR A BLANK OR COLON
07C2 C2C20C            JNZ     ERRL    ;ERROR IF NOT A BLANK
07C5 C31E07            JMP     OPC
               ;
               ;
               ;
               ; PROCESS PSEUDO OPS FOR PASS2
07C8 1A        PSU2:   LDAX    D
07C9 B7                ORA     A       ;SET FLAGS
07CA CA0C08            JZ      ORG2    ;ORG OPCODE
07CD FAF107            JM      DAT2    ;DATA OPCODE
07D0 E2FA07            JPO     EQU2    ;EQUATE PSEUDE-OP
07D3 FE05              CPI     5
07D5 DAE107            JC      RES2    ;RES OPCODE
07D8 C20109            JNZ     EASS    ;END OPCODE
               ; DO DW PSEUDO-OP
07DB CDE108    ACO2:   CALL    TYS6    ;GET VALUE
07DE C35A07            JMP     ACO1
               ; DO DS PSEUDO-OP
07E1 CD940B    RES2:   CALL    ASBL    ;GET OPERAND
07E4 44                MOV     B,H
07E5 4D                MOV     C,L
07E6 2A8C10            LHLD    BBUF+2  ;FETCH STORAGE COUNTER
07E9 09                DAD     B       ;ADD VALUE
07EA 228C10            SHLD    BBUF+2
07ED AF        RES21:  XRA     A       ;GET A ZERO
07EE C3F80A            JMP     OCN2
               ; DO DB PSEUDO-OP
07F1 CDA008    DAT2:   CALL    TYS5    ;GET OPERAND
07F4 AF        DAT2A:  XRA     A       ;MAKE A ZERO
07F5 0E01              MVI     C,1     ;BYTE COUNT
07F7 C3F50A            JMP     OCN1
               ;
               ; HANDLE EQUATES ON 2ND PASS.
               ;
07FA CD940B    EQU2:   CALL    ASBL    ;GET OPERAND INTO HL AND
                                       ;  FALL INTO NEXT ROUTINE
               ;
               ; STORE CONTENTS OF HL AS HEX ASCII AT OBUF+2.
               ;   ON RETURN, DE HOLDS VALUE WHICH WAS IN HL.
               ;
07FD EB        BINAD:  XCHG            ;PUT VALUE INTO DE
07FE 218410            LXI     H,OBUF+2 ;POINTER TO ADDR IN OBUF
0801 7A                MOV     A,D     ;STORE HI BYTE....
0802 CD8902            CALL    BINH+3
0805 23                INX     H
0806 7B                MOV     A,E     ;STORE LO BYTE...
0807 CD8902            CALL    BINH+3
080A 23                INX     H
080B C9                RET
```

```
                ; DO ORG PSEUDO-OP
080C CD9408     ORG2:    CALL     ASBL      ;GET NEW ORIGIN
080F 3A8210              LDA      OBUF      ;GET ERROR INDICATOR
0812 FE20                CPI      ' '       ;CHECK FOR AN ERROR
0814 C0                  RNZ                ;DON'T MODIFY PC IF ERROR
0815 CDFD07              CALL     BINAD     ;STORE NEW ADDR IN OBUF
0818 2A9210              LHLD     ASPC      ;FETCH PC
081B EB                  XCHG
081C 229210              SHLD     ASPC      ;STORE NEW PC
081F 7D                  MOV      A,L
0820 93                  SUB      E         ;FORM DIFFERENCE OF ORIGINS
0821 5F                  MOV      E,A
0822 7C                  MOV      A,H
0823 9A                  SBB      D
0824 57                  MOV      D,A
0825 2A8C10              LHLD     BBUF+2    ;FETCH STORAGE POINTER
0828 19                  DAD      D         ;MODIFY
0829 228C10              SHLD     BBUF+2    ;SAVE
082C C9                  RET
                ;
                ; PROCESS 1 BYTE INSTRUCTIONS WITHOUT OPERANDS
                ;
082D CDEE08     TYP1:    CALL     ASTO      ;STORE VALUE IN MEMORY
0830 C9                  RET
                ;
                ; PROCESS STAX AND LDAX INSTRUCTIONS
                ;
0831 CD9408     TYP2:    CALL     ASBL      ;FETCH OPERAND
0834 C4810C              CNZ      ERRR      ;ILLEGAL REGISTER
0837 7D                  MOV      A,L       ;GET LOW ORDER OPERAND
0838 B7                  ORA      A         ;SET FLAGS
0839 CA5508              JZ       TY31      ;OPERAND = 0
083C FE02                CPI      2         ;OPERAND = 2
083E C4810C              CNZ      ERRR      ;ILLEGAL REGISTER
0841 C35508              JMP      TY31
                ;
                ; PROCESS PUSH,POP,INX,DCX,DAD INSTRUCTIONS
                ;
0844 CD9408     TYP3:    CALL     ASBL      ;FETCH OPERAND
0847 C4810C              CNZ      ERRR      ;ILLEGAL REGISTER
084A 7D                  MOV      A,L       ;GET LOW ORDER OPERAND
084B 0F                  RRC                ;CHECK LOW ORDER BIT
084C DC810C              CC       ERRR      ;ILLEGAL REGISTER
084F 17                  RAL                ;RESTORE
0850 FE08                CPI      8
0852 D4810C              CNC      ERRR      ;ILLEGAL REGISTER
0855 07         TY31:    RLC                ;MULTIPLY BY 8
0856 17                  RAL
0857 17                  RAL
0858 47         TY32:    MOV      B,A
0859 1A                  LDAX     D         ;FETCH OPCODE BASE
085A 80                  ADD      B         ;FORM OPCODE
085B FE76                CPI      118       ;CHECK FOR MOV M,M
085D CC810C              CZ       ERRR      ;ILLEGAL REGISTER
0860 C32D08              JMP      TYP1
                ;
                ; PROCESS ACCUMULATOR, INR,DCR,MOV,RST INSTRUCTIONS
                ;
0863 CD9408     TYP4:    CALL     ASBL      ;FETCH OPERAND
0866 C4810C              CNZ      ERRR      ;ILLEGAL REGISTER
0869 7D                  MOV      A,L       ;GET LOW ORDER OPERAND
086A FE08                CPI      8
086C D4810C              CNC      ERRR      ;ILLEGAL REGISTER
086F 1A                  LDAX     D         ;FETCH OPCODE BASE
0870 FE40                CPI      64        ;CHECK FOR MOV INSTRUCTION
0872 CA8108              JZ       TY41
```

```
0875 FEC7              CPI      199
0877 7D                MOV      A,L
0878 CA5508            JZ       TY31      ;RST INSRUCTION
087B FA5808            JM       TY32      ;ACCUMULATOR INSTRUCTION
087E C35508            JMP      TY31      ;INR,DCR
              ; PROCESS MOV INSTRUCTION
0881 29       TY41:    DAD      H         ;MULTIPLY OPERAND BY 8
0882 29                DAD      H
0883 29                DAD      H
0884 85                ADD      L         ;FORM OPCODE
0885 12                STAX     D         ;SAVE OPCODE
0886 CD8F08            CALL     MPNT
0889 CD970B            CALL     ASCN
088C C4810C            CNZ      ERRR      ;INCREMENT POINTER
088F 7D                MOV      A,L       ;FETCH LOW ORDER OPERAND
0890 FE08              CPI      8
0892 D4810C            CNC      ERRR      ;ILLEGAL REGISTER
0895 C35808            JMP      TY32
              ;
              ; PROCESS IMMEDIATE INSTRUCTIONS
              ; IMMEDIATE BYTE CAN BETWEEN -256 AND +255
              ; MVI INSTRUCTION IS A SPECIAL CASE AND CONTAINS
              ; 2 ARGUMENTS IN OPERAND
0898 FE06     TYP5:    CPI      6         ;CHECK FOR MVI INSTRUCTION
089A CCAD08            CZ       TY56
089D CDEE08            CALL     ASTO      ;STORE OBJECT BYTE
08A0 CD940B   TY55:    CALL     ASBL      ;GET IMMEDIATE ARGUMENT
08A3 3C                INR      A
08A4 FE02              CPI      2         ;CHECK OPERAND FOR RANGE
08A6 D49A0C            CNC      ERRV      ;OPERAND OUT OF RANGE
08A9 7D                MOV      A,L
08AA C32D08            JMP      TYP1
              ;
              ; FETCH 1ST ARG FOR MVI AND LXI INSTRUCTIONS
              ;
08AD CD940B   TY56:    CALL     ASBL      ;FETCH ARG
08B0 C4810C            CNZ      ERRR      ;ILLEGAL REGISTER
08B3 7D                MOV      A,L       ;GET LOW ORDER ARGUMENT
08B4 FE08              CPI      8
08B6 D4810C            CNC      ERRR      ;ILLEGAL REGISTER
08B9 29                DAD      H
08BA 29                DAD      H
08BB 29                DAD      H
08BC 1A                LDAX     D         ;FETCH OPCODE BASE
08BD 85                ADD      L         ;FOR OPCODE
08BE 5F                MOV      E,A       ;SAVE OBJECT BYTE
08BF 2A9610   MPNT:    LHLD     PNTR      ;FETCH POINTER
08C2 7E                MOV      A,M       ;FETCH CHARACTER
08C3 FE2C              CPI      ','       ;CHECK FOR COMMA
08C5 23                INX      H         ;INCREMENT POINTER
08C6 229610            SHLD     PNTR
08C9 C28A0C            JNZ      ERRS      ;SYNTAX ERROR IF NO COMMA
08CC 7B                MOV      A,E
08CD C9                RET
              ;
              ; PROCESS 3 BYTE INSTRUCTIONS
              ; LXI INSTRUCTION IS A SPECIAL CASE
              ;
08CE FE01     TYP6:    CPI      1         ;CHECK FOR LXI INSTRUCTION
08D0 C2DE08            JNZ      TY6       ;JUMP IF NOT LXI
08D3 CDAD08            CALL     TY56      ;GET REGISTER
08D6 E608              ANI      08H       ;CHECK FOR ILLEGAL REGISTER
08D8 C4810C            CNZ      ERRR      ;REGISTER ERROR
08DB 7B                MOV      A,E       ;GET OPCODE
08DC E6F7              ANI      0F7H      ;CLEAR BIT IN ERROR
08DE CDEE08   TY6:     CALL     ASTO      ;STORE OBJECT BYTE
```

```
08E1 CD940B    TYS6:    CALL     ASBL     ;FETCH OPERAND
08E4 7D                 MOV      A,L
08E5 54                 MOV      D,H
08E6 CDEE08             CALL     ASTO     ;STORE 2ND BYTE
08E9 7A                 MOV      A,D
08EA C32D08             JMP      TYP1
08ED C9                 RET
               ;
               ; THIS ROUTINE IS USED TO STORE OBJECT CODE PRODUCED
               ; BY THE ASSEMBLER DURING PASS 2 INTO MEMORY
               ;
08EE 2A8C10    ASTO:    LHLD     BBUF+2   ;FETCH STORAGE ADDRESS
08F1 77                 MOV      M,A      ;STORE OBJECT BYTE
08F2 23                 INX      H        ;INCREMENT LOCATION
08F3 228C10             SHLD     BBUF+2
08F6 2A9E10             LHLD     OIND     ;FETCH OUTPUT ADDRESS
08F9 23                 INX      H
08FA CD8902             CALL     BINH+3   ;CONVERT OBJECT BYTE
08FD 229E10             SHLD     OIND
0900 C9                 RET
               ;
               ; GET HERE WHEN END PSEUDO-OP IS FOUND OR WHEN
               ; END-OF-FILE OCCURS IN SOURCE FILE. CONTROL IS SET
               ; FOR EITHER PASS 2 OR ASSEMBLY TERMINATOR IF FINISHED.
               ;
0901 3A9410    EASS:    LDA      PASI     ;FETCH PASS INDICATOR
0904 B7                 ORA      A        ;SET FLAGS
0905 C26700             JNZ      EOR      ;JUMP IF FINISHED
0908 3E01               MVI      A,1      ;PASS INDICATOR FOR 2ND PASS
090A C37A06             JMP      ASM3     ;DO 2ND PASS
               ;
               ; THIS ROUTINE SCANS THROUGH A CHARACTER STRING UNTIL
               ; THE FIRST NON-BLANK CHARACTER IS FOUND
               ;
               ; ON RETURN, CARRY SET INDICATES A CARRIAGE RETURN
               ; AS FIRST NON-BLANK CHARACTER.
               ;
090D 2A9610    SBLK:    LHLD     PNTR     ;FETCH ADDRESS
0910 7E        SBL1:    MOV      A,M      ;FETCH CHARACTER
0911 FE20               CPI      ' '      ;CHECK FOR A BLANK
0913 C0                 RNZ               ;RETURN IF NON-BLANK
0914 23        SBL2:    INX      H        ;INCREMENT
0915 229610             SHLD     PNTR     ;SAVE POINTER
0918 C31009             JMP      SBL1
               ;
               ;
               ; THIS ROUTINE IS USED TO CHECK THE CONDITION
               ; CODE MNEMONICS FOR CONDITIONAL JUMPS, CALLS,
               ; AND RETURNS.
               ;
091B 217F10    COND:    LXI      H,ABUF+1
091E 227410             SHLD     ADDS
0921 0602               MVI      B,2      ;2 CHARACTERS
0923 CD500A             CALL     COPC
0926 C9                 RET
               ;
               ;
               ; THE FOLLOWING IS THE OPCODE TABLE
               ;
               ;
0927 4F5247    OTAB:    DB       'ORG'
092A 00                 DB       0
092B 00                 DB       0
092C 455155             DB       'EQU'
092F 00                 DB       0
0930 01                 DB       1
```

```
0931 4442                DB        'DB'
0933 00                  DB        0
0934 00                  DB        0
0935 FF                  DB        -1
0936 4453                DB        'DS'
0938 00                  DB        0
0939 00                  DB        0
093A 03                  DB        3
093B 4457                DB        'DW'
093D 00                  DB        0
093E 00                  DB        0
093F 05                  DB        5
0940 454E44              DB        'END'
0943 00                  DB        0
0944 06                  DB        6
0945 00                  DB        0
0946 484C54              DB        'HLT'
0949 76                  DB        118
094A 524C43              DB        'RLC'
094D 07                  DB        7
094E 525243              DB        'RRC'
0951 0F                  DB        15
0952 52414C              DB        'RAL'
0955 17                  DB        23
0956 524152              DB        'RAR'
0959 1F                  DB        31
095A 524554              DB        'RET'
095D C9                  DB        201
095E 434D41              DB        'CMA'
0961 2F                  DB        47
0962 535443              DB        'STC'
0965 37                  DB        55
0966 444141              DB        'DAA'
0969 27                  DB        39
096A 434D43              DB        'CMC'
096D 3F                  DB        63
096E 4549                DB        'EI'
0970 00                  DB        0
0971 FB                  DB        251
0972 4449                DB        'DI'
0974 00                  DB        0
0975 F3                  DB        243
0976 4E4F50              DB        'NOP'
0979 00                  DB        0
097A 00                  DB        0
097B 58434847            DB        'XCHG'
097F EB                  DB        235
0980 5854484C            DB        'XTHL'
0984 E3                  DB        227
0985 5350484C            DB        'SPHL'
0989 F9                  DB        249
098A 5043484C            DB        'PCHL'
098E E9                  DB        233
098F 00                  DB        0
0990 53544158            DB        'STAX'
0994 02                  DB        2
0995 4C444158            DB        'LDAX'
0999 0A                  DB        10
099A 00                  DB        0
099B 50555348            DB        'PUSH'
099F C5                  DB        197
09A0 504F50              DB        'POP'
09A3 00                  DB        0
09A4 C1                  DB        193
09A5 494E58              DB        'INX'
09A8 00                  DB        0
```

```
09A9 03                 DB       3
09AA 444358             DB       'DCX'
09AD 00                 DB       0
09AE 0B                 DB       11
09AF 444144             DB       'DAD'
09B2 00                 DB       0
09B3 09                 DB       9
09B4 00                 DB       0
09B5 494E52             DB       'INR'
09B8 04                 DB       4
09B9 444352             DB       'DCR'
09BC 05                 DB       5
09BD 4D4F56             DB       'MOV'
09C0 40                 DB       64
09C1 414444             DB       'ADD'
09C4 80                 DB       128
09C5 414443             DB       'ADC'
09C8 88                 DB       136
09C9 535542             DB       'SUB'
09CC 90                 DB       144
09CD 534242             DB       'SBB'
09D0 98                 DB       152
09D1 414E41             DB       'ANA'
09D4 A0                 DB       160
09D5 585241             DB       'XRA'
09D8 A8                 DB       168
09D9 4F5241             DB       'ORA'
09DC B0                 DB       176
09DD 434D50             DB       'CMP'
09E0 B8                 DB       184
09E1 525354             DB       'RST'
09E4 C7                 DB       199
09E5 00                 DB       0
09E6 414449             DB       'ADI'
09E9 C6                 DB       198
09EA 414349             DB       'ACI'
09ED CE                 DB       206
09EE 535549             DB       'SUI'
09F1 D6                 DB       214
09F2 534249             DB       'SBI'
09F5 DE                 DB       222
09F6 414E49             DB       'ANI'
09F9 E6                 DB       230
09FA 585249             DB       'XRI'
09FD EE                 DB       238
09FE 4F5249             DB       'ORI'
0A01 F6                 DB       246
0A02 435049             DB       'CPI'
0A05 FE                 DB       254
0A06 494E               DB       'IN'
0A08 00                 DB       0
0A09 DB                 DB       219
0A0A 4F5554             DB       'OUT'
0A0D D3                 DB       211
0A0E 4D5649             DB       'MVI'
0A11 06                 DB       6
0A12 00                 DB       0
0A13 4A4D50             DB       'JMP'
0A16 00                 DB       0
0A17 C3                 DB       195
0A18 43414C4C           DB       'CALL'
0A1C CD                 DB       205
0A1D 4C5849             DB       'LXI'
0A20 00                 DB       0
0A21 01                 DB       1
0A22 4C4441             DB       'LDA'
```

```
0A25 00                 DB       0
0A26 3A                 DB       58
0A27 535441             DB       'STA'
0A2A 00                 DB       0
0A2B 32                 DB       50
0A2C 53484C44           DB       'SHLD'
0A30 22                 DB       34
0A31 4C484C44           DB       'LHLD'
0A35 2A                 DB       42
0A36 00                 DB       0
               ; CONDITION CODE TABLE
0A37 4E5A               DB       'NZ'
0A39 00                 DB       0
0A3A 5A                 DB       'Z'
0A3B 00                 DB       0
0A3C 08                 DB       8
0A3D 4E43               DB       'NC'
0A3F 10                 DB       16
0A40 43                 DB       'C'
0A41 00                 DB       0
0A42 18                 DB       24
0A43 504F               DB       'PO'
0A45 20                 DB       32
0A46 5045               DB       'PE'
0A48 28                 DB       40
0A49 50                 DB       'P'
0A4A 00                 DB       0
0A4B 30                 DB       48
0A4C 4D                 DB       'M'
0A4D 00                 DB       0
0A4E 38                 DB       56
0A4F 00                 DB       0
               ;
               ; THIS ROUTINE IS USED TO CHECK A GIVEN OPCODE
               ; AGAINST THE LEGAL OPCODES IN THE OPCODE TABLE.
               ;
0A50 2A7410    COPC:    LHLD     ADDS
0A53 1A                 LDAX     D         ;FETCH CHARACTER
0A54 B7                 ORA      A         ;SET FLAGS
0A55 CA620A             JZ       COP1      ;JUMP IF TERMINATION CHARACTER
0A58 48                 MOV      C,B
0A59 CD5301             CALL     SEAR
0A5C 1A                 LDAX     D
0A5D C8                 RZ                 ;RETURN IF MATCH
0A5E 13                 INX      D         ; NEXT STRING
0A5F C3500A             JMP      COPC      ;CONTINUE SEARCH
0A62 3C        COP1:    INR      A         ;CLEAR ZERO FLAG
0A63 13                 INX      D         ;INCREMENT ADDRESS
0A64 C9                 RET
               ;
               ; THIS ROUTINE CHECKS THE LEGAL OPCODES IN BOTH PASS 1
               ; AND PASS 2.  IN PASS 1 THE PROGRAM COUNTER IS INCRE-
               ; MENTED BY THE CORRECT NUMBER OF BYTES.  AN ADDRESS IS
               ; ALSO SET SO THAT AN INDEXED JUMP CAN BE MADE TO
               ; PROCESS THE OPCODE FOR PASS 2.
               ;
               ;
0A65 217E10    OPCD:    LXI      H,ABUF    ;GET ADDRESS
0A68 227410             SHLD     ADDS
0A6B 112709             LXI      D,OTAB    ;OPCODE TABLE ADDRESS
0A6E 0604               MVI      B,4       ;CHARACTER COUNT
0A70 CD500A             CALL     COPC      ;CHECK OPCODES
0A73 CA0E0B             JZ       PSEU      ;JUMP IF A PSEUDO-OP
0A76 05                 DCR      B         ;3 CHARACTER OPCODES
0A77 CD500A             CALL     COPC
0A7A CA810A             JZ       OP1
```

```
0A7D 04                 INR     B       ;4 CHARACTER  OPCODES
0A7E CD500A             CALL    COPC
0A81 212D08     OP1:    LXI     H,TYP1  ;TYPE 1 INSTRUCTIONS
0A84 0E01       OP2:    MVI     C,1     ;1 BYTE INSTRUCTIONS
0A86 CAE10A             JZ      OCNT
                ;
0A89 CD500A     OPC2:   CALL    COPC    ;CHECK FOR STAX,LDAX
0A8C 213108             LXI     H,TYP2
0A8F CA840A             JZ      OP2
0A92 CD500A             CALL    COPC    ;CHECK FOR PUSH,POP,INX
                ;               ;       ;DCX AND DAD
0A95 214408             LXI     H,TYP3
0A98 CA840A             JZ      OP2
0A9B 05                 DCR     B       ;3 CHAR OPCODES
0A9C CD500A             CALL    COPC    ;ACCUMULATOR INSTRUCTIONS,
                ;               ;       ;INR,DCR,MOV,RST
0A9F 216308             LXI     H,TYP4
0AA2 CA840A             JZ      OP2
                ;
0AA5 CD500A     OPC3:   CALL    COPC    ;IMMEDIATE INSTRUCTIONS
0AA8 219808             LXI     H,TYP5
0AAB 0E02               MVI     C,2     ;2 BYTE INSTRUCTIONS
0AAD CAE10A             JZ      OCNT
0AB0 04                 INR     B       ;4 CHAR OPCODES
0AB1 CD500A             CALL    COPC    ;JMP,CALL,LXI,LDA,STA,
                ;               ;       ;LHLD,SHLD OPCODES
0AB4 CADC0A             JZ      OP4
0AB7 CD1B09             CALL    COND    ;CONDITIONAL INSTRUCTIONS
0ABA C2060B             JNZ     OERR    ;ILLEGAL OPCODE
0ABD C6C0               ADI     192     ;ADD BASE VALUE TO RETURN
0ABF 57                 MOV     D,A
0AC0 0603               MVI     B,3     ;3 CHARACTER OPCODES
0AC2 3A7E10             LDA     ABUF    ;FETCH FIRST CHARACTER
0AC5 4F                 MOV     C,A     ;SAVE CHARACTER
0AC6 FE52               CPI     'R'     ;CONDITIONAL RETURN
0AC8 7A                 MOV     A,D
0AC9 CA810A             JZ      OP1
0ACC 79                 MOV     A,C
0ACD 14                 INR     D       ;FORM CONDITIONAL JUMP
0ACE 14                 INR     D
0ACF FE4A               CPI     'J'     ;CONDITIONAL JUMP
0AD1 CADB0A             JZ      OPAD
0AD4 FE43               CPI     'C'     ;CONDITIONAL CALL
0AD6 C2060B             JNZ     OERR    ;ILLEGAL OPCODE
0AD9 14                 INR     D       ;FORM CONDIITIONAL CALL
0ADA 14                 INR     D
0ADB 7A         OPAD:   MOV     A,D     ;GET OPCODE
0ADC 21CE08     OP4:    LXI     H,TYP6
0ADF 0E03       OP5:    MVI     C,3     ;3 BYTE INSTRUCTION
0AE1 329D10     OCNT:   STA     TEMP    ;SAVE OPCODE
                ;
                ; CHECK FOR OPCODE ONLY CONTAINING THE CORRECT NUMBER OF
                ; CHARACTERS. THUS ADDQ, SAY, WOULD GIVE AN ERROR
                ;
0AE4 3E7E               MVI     A,ABUF AND 0FFH ;LOAD BUFFER ADDRESS
0AE6 80                 ADD     B       ;ADD LENGTH OF OPCODE
0AE7 5F                 MOV     E,A
0AE8 3E10               MVI     A,ABUF/256
0AEA CE00               ACI     0       ;GET HIGH ORDER ADDRESS
0AEC 57                 MOV     D,A
0AED 1A                 LDAX    D       ;FETCH CHARACTER AFTER OPCODE
0AEE B7                 ORA     A       ;IT SHOULD BE ZERO
0AEF C2060B             JNZ     OERR    ;OPCODE ERROR
0AF2 3A9410             LDA     PASI    ;FETCH PASS INDICATOR
0AF5 0600       OCN1:   MVI     B,0
0AF7 EB                 XCHG
```

```
0AF8 2A9210   OCN2:  LHLD   ASPC     ;FETCH PROGRAM COUNTER
0AFB 09              DAD    B        ;ADD IN BYTE COUNT
0AFC 229210          SHLD   ASPC     ;STORE PC
0AFF B7              ORA    A        ;WHICH PASS?
0B00 C8              RZ              ;RETURN IF PASS 1
0B01 3A9D10          LDA    TEMP     ;FETCH OPCODE
0B04 EB              XCHG
0B05 E9              PCHL
              ;
0B06 21AD0C   OERR:  LXI    H,ERRO   ;SET ERROR ADDRESS
0B09 0E03            MVI    C,3      ;LEAVE 3 BYTES FOR PATCH
0B0B C3F20A          JMP    OCN1-3
              ;
0B0E 218210   PSEU:  LXI    H,ABUF+4         ;SET BUFFER ADDRESS
0B11 7E              MOV    A,M      ;FETCH CHARACTER AFTER OPCODE
0B12 B7              ORA    A        ;SHOULD BE A ZERO
0B13 C2060B          JNZ    OERR
0B16 3A9410          LDA    PASI     ;FETCH PASS INDICATOR
0B19 B7              ORA    A
0B1A CA4407          JZ     PSU1
0B1D C3C807          JMP    PSU2
              ;
              ;
              ; THIS ROUTINE IS USED TO PROCESS LABELS.
              ; IT CHECKS TO SEE IF A LABEL IS IN THE SYMBOL TABLE
              ; OR NOT.  ON RETURN, Z=1 INDICATES A MATCH WAS FOUND
              ; AND H,L CONTAIN THE VALUE ASSOCIATED WITH THE LABEL.
              ; THE REGISTER NAMES A, B, C, D, E, H, L, P, AND S ARE
              ; PRE-DEFINED AND NEED NOT BE ENTERED BY THE USER.
              ; ON RETURN, C=1 INDICATES A LABEL ERROR.
              ;
0B20 FE41     SLAB:  CPI    'A'      ;CHECK FOR LEGAL CHAR
0B22 D8              RC
0B23 FE5B            CPI    'Z'+1    ;CHECK FOR ILLEGAL CHAR
0B25 3F              CMC
0B26 D8              RC              ;RETURN IF ILLEGAL CHAR
0B27 CD750B          CALL   ALPS     ;PLACE SYMBOL IN BUFFER
0B2A 217E10          LXI    H,ABUF   ;SET BUFFER ADDRESS
0B2D 227410          SHLD   ADDS     ;SAVE ADDRESS
0B30 05              DCR    B        ;CHECK IF ONE CHARACTER
0B31 C2440B          JNZ    SLA1
              ; CHECK IF PREDEFINED REGISTER NAME
0B34 04              INR    B        ;SET B=1
0B35 11600B          LXI    D,RTAB   ;REGISTER TABLE ADDRESS
0B38 CD500A          CALL   COPC     ;CHECK NAME OF REGISTER
0B3B C2440B          JNZ    SLA1     ;NOT A PREDEFINED REGISTER
0B3E 6F              MOV    L,A      ;SET VALUE(HIGH)
0B3F 2600            MVI    H,0
0B41 C35A0B          JMP    SLA2
0B44 3A9810   SLA1:  LDA    NOLA     ;FETCH SYMBOL COUNT
0B47 47              MOV    B,A
0B48 111A11          LXI    D,SYMT   ;SET SYMBOL TABLE ADDRESS
0B4B B7              ORA    A        ;ARE THERE ANY LABELS?
0B4C CA5D0B          JZ     SLA3     ;JUMP IF NO LABELS
0B4F 3E05            MVI    A,LLAB   ;FETCH LENGTH OF LABEL
0B51 329510          STA    NCHR
0B54 CD3C01          CALL   COMS     ;CHECK TABEL
0B57 4C              MOV    C,H      ;SWAP H AND L
0B58 65              MOV    H,L
0B59 69              MOV    L,C
0B5A 37       SLA2:  STC             ;SET CARRY
0B5B 3F              CMC             ;CLEAR CARRY
0B5C C9              RET             ;RETURN
0B5D 3C       SLA3:  INR    A        ;CLEAR ZERO FLAG
0B5E B7              ORA    A        ;CLEAR CARRY
0B5F C9              RET
```

```
                ;
                ; PREDEFINE REGISTER VALUES IN THIS TABLE
                ;
0B60 41         RTAB:   DB      'A'
0B61 07                 DB      7
0B62 42                 DB      'B'
0B63 00                 DB      0
0B64 43                 DB      'C'
0B65 01                 DB      1
0B66 44                 DB      'D'
0B67 02                 DB      2
0B68 45                 DB      'E'
0B69 03                 DB      3
0B6A 48                 DB      'H'
0B6B 04                 DB      4
0B6C 4C                 DB      'L'
0B6D 05                 DB      5
0B6E 4D                 DB      'M'
0B6F 06                 DB      6
0B70 50                 DB      'P'
0B71 06                 DB      6
0B72 53                 DB      'S'
0B73 06                 DB      6
0B74 00                 DB      0          ;END OF TABLE INDICATOR
                ;
                ; THIS ROUTINE SCANS THE INPUT LINE AND PLACES THE
                ; OPCODES AND LABELS IN THE BUFFER.  THE SCAN TERMINATES
                ; WHEN A CHARACTER OTHER THAN 0-9 OR A-Z IS FOUND.
                ;
0B75 0600       ALPS:   MVI     B,0        ;SET COUNT
0B77 12         ALP1:   STAX    D          ;STORE CHARACTER IN BUFFER
0B78 04                 INR     B          ;INCREMENT COUNT
0B79 78                 MOV     A,B        ;FETCH COUNT
0B7A FE08               CPI     11         ;MAXIMUM BUFFER SIZE
0B7C D0                 RNC                ;RETURN IF BUFFER FILLED
0B7D 13                 INX     D          ;INCREMENT BUFFER
0B7E 23                 INX     H          ;INCREMENT INPUT POINTER
0B7F 229610             SHLD    PNTR       ;SAVE LINE POINTER
0B82 7E                 MOV     A,M        ;FETCH CHARACTER
0B83 FE30               CPI     '0'        ;CHECK FOR LEGAL CHARACTERS
0B85 D8                 RC
0B86 FE3A               CPI     '9'+1
0B88 DA7708             JC      ALP1
0B8B FE41               CPI     'A'
0B8D D8                 RC
0B8E FE5B               CPI     'Z'+1
0B90 DA7708             JC      ALP1
0B93 C9                 RET
                ; THIS ROUTINE IS USED TO SCAN THROUGH THE INPUT LINE
                ; TO FETCH THE VALUE OF THE OPERAND FIELD.  ON RETURN,
                ; THE VALUE OF THE OPERAND IS CONTAINED IN REG'S H,L.
                ;
0B94 CD0D09     ASBL:   CALL    SBLK       ;GET FIRST ARGUMENT
0B97 210000     ASCN:   LXI     H,0        ;GET A ZERO
0B9A 229A10             SHLD    OPRD       ;INITIALIZE OPERAND
0B9D 24                 INR     H
0B9E 229B10             SHLD    OPRI-1     ;INITIALIZE OPERAND INDICATOR
0BA1 2A9610     NXT1:   LHLD    PNTR       ;FETCH SCAN POINTER
0BA4 2B                 DCX     H
0BA5 CD6601             CALL    ZBUF       ;CLEAR BUFFER
0BA8 329910             STA     SIGN       ;ZERO SIGN INDICATOR
0BAB 23         NXT2:   INX     H          ;INCREMENT POINTER
0BAC 7E                 MOV     A,M        ;FETCH NEXT CHARACTER
0BAD FE21               CPI     ' '+1
0BAF DA530C             JC      SEND       ;JUMP IF CR OR BLANK
0BB2 FE2C               CPI     ','        ;FIELD SEPARATOR
```

```
0B84 CA530C                JZ      SEND
                   ; CHECK FOR OPERATORS
0B87 FE2B                  CPI     '+'       ;CHECK FOR PLUS
0B89 CAC40B                JZ      ASC1
0B8C FE2D                  CPI     '-'       ;CHECK FOR MINUS
0B8E C2D40B                JNZ     ASC2
0BC1 329910                STA     SIGN
0BC4 3A9C10        ASC1:   LDA     OPRI      ;FETCH OPERAND INDICATOR
0BC7 FE02                  CPI     2         ;CHECK FOR TWO OPERATORS
0BC9 CA8A0C                JZ      ERRS      ;SYNTAX ERROR
0BCC 3E02                  MVI     A,2
0BCE 329C10                STA     OPRI      ;SET INDICATOR
0BD1 C3A80B                JMP     NXT2
                   ;CHECK FOR OPERANDS
0BD4 4F            ASC2:   MOV     C,A       ;SAVE CHARACTER
0BD5 3A9C10                LDA     OPRI      ;GET INDICATOR
0BD8 B7                    ORA     A         ;CHECK FOR TWO OPERANDS
0BD9 CA8A0C                JZ      ERRS      ;SYNTAX ERROR
0BDC 79                    MOV     A,C
0BDD FE24                  CPI     '$'       ;LC EXPRESSION
0BDF C2EC0B                JNZ     ASC3
0BE2 23                    INX     H         ;INCREMENT POINTER
0BE3 229610                SHLD    PNTR      ;SAVE POINTER
0BE6 2A9210                LHLD    ASPC      ;FETCH LOCATION COUNTER
0BE9 C3280C                JMP     AVAL
                   ;CHECK FOR ASCII CHARACTERS
0BEC FE27          ASC3:   CPI     27H       ;CHECK FOR SINGLE QUOTE
0BEE C2180C                JNZ     ASC5      ;JUMP IF NOT QUOTE
0BF1 110000                LXI     D,0       ;GET A ZERO
0BF4 0E03                  MVI     C,3       ;CHARACTER COUNT
0BF6 23            ASC4:   INX     H         ;BUMP POINTER
0BF7 229610                SHLD    PNTR      ;SAVE
0BFA 7E                    MOV     A,M       ;FETCH NEXT CHARACTER
0BFB FE0D                  CPI     ASCR      ;IS IT A CR?
0BFD CAA80C                JZ      ERRA      ;ARGUMENT ERROR
0C00 FE27                  CPI     27H       ;IS IT QUOTE
0C02 C20F0C                JNZ     SSTR
0C05 23                    INX     H         ;INCREMENT POINTER
0C06 229610                SHLD    PNTR      ;SAVE
0C09 7E                    MOV     A,M       ;FETCH NEXT CHAR
0C0A FE27                  CPI     27H       ;CHECK FOR 2 QUOTES IN A ROW
0C0C C2290C                JNZ     AVAL+1    ;TERMINAL QUOTE
0C0F 0D            SSTR:   DCR     C         ;CHECK COUNT
0C10 CAA80C                JZ      ERRA      ;TOO MANY CHARACTERS
0C13 53                    MOV     D,E
0C14 5F                    MOV     E,A       ;SET CHARACTER IN BUFFER
0C15 C3F60B                JMP     ASC4
0C18 FE30          ASC5:   CPI     '0'       ;CHECK FOR NUMERIC
0C1A DAA80C                JC      ERRA      ;ILLEGAL CHARACTER
0C1D FE3A                  CPI     '9'+1
0C1F D2470C                JNC     ALAB
0C22 CD630C                CALL    NUMS      ;GET NUMERIC VALUE
0C25 DAA80C                JC      ERRA      ;ARGUMENT ERROR
0C28 EB            AVAL:   XCHG
0C29 2A9A10                LHLD    OPRD      ;FETCH OPERAND
0C2C AF                    XRA     A         ;GET A ZERO
0C2D 329C10                STA     OPRI      ;STOR IN OPERAND INDICATOR
0C30 3A9910                LDA     SIGN      ;GET SIGN INDICATOR
0C33 B7                    ORA     A         ;SET FLAGS
0C34 C23E0C                JNZ     ASUB
0C37 19                    DAD     D         ;FORM RESULT
0C38 229A10        ASC7:   SHLD    OPRD      ;SAVE RESULT
0C3B C3A10B                JMP     NXT1
0C3E 7D            ASUB:   MOV     A,L
0C3F 93                    SUB     E
0C40 6F                    MOV     L.A
```

```
0C41 7C                 MOV     A,H
0C42 9A                 SBB     D
0C43 67                 MOV     H,A
0C44 C3380C             JMP     ASC7
0C47 CD2008     ALAB:   CALL    SLAB
0C4A CA280C             JZ      AVAL
0C4D DAA80C             JC      ERRA     ;ILLEGAL SYMBOL
0C50 C3950C             JMP     ERRU     ;UNDEFINED SYMBOL
                ;
                ; GET HERE WHEN TERMINATING CHARACTER IS FOUND.
                ; CHECK FOR LEADING FIELD SEPARATOR.
                ;
0C53 3A9C10     SEND:   LDA     OPRI     ;FETCH OPERAND INDICATOR
0C56 B7                 ORA     A        ;SET FLAGS
0C57 C28A0C             JNZ     ERRS     ;SYNTAX ERROR
0C5A 2A9A10             LHLD    OPRD
0C5D 7C         SEN1:   MOV     A,H      ;GET HIGH ORDER BYTE
0C5E 119D10             LXI     D,TEMP   ;GET ADDRESS
0C61 B7                 ORA     A        ;SET FLAGS
0C62 C9                 RET
                ;
                ; GET A NUMERIC VALUE WHICH IS EITHER HEXADECIMAL OR
                ; DECIMAL.  ON RETURN, CARRY SET INDICATES AN ERROR.
                ;
0C63 CD750B     NUMS:   CALL    ALPS     ;GET NUMERIC
0C66 1B                 DCX     D
0C67 1A                 LDAX    D        ;GET LAST CHARACTER
0C68 017E10             LXI     B,ABUF   ;SET BUFFER ADDRESS
0C6B FE48               CPI     'H'      ;IS IT HEXADECIMAL?
0C6D CA7B0C             JZ      NUM2
0C70 FE44               CPI     'D'      ;IS IT DECIMAL
0C72 C2770C             JNZ     NUM1
0C75 AF                 XRA     A        ;GET A ZERO
0C76 12                 STAX    D        ;CLEAR D FROM BUFFER
0C77 CD0102     NUM1:   CALL    ADEC     ;CONVERT DECIMAL VALUE
0C7A C9                 RET
0C7B AF         NUM2:   XRA     A        ;GET A ZERO
0C7C 12                 STAX    D        ;CLEAR H FROM BUFFER
0C7D CD1B02             CALL    AHEX
0C80 C9                 RET
                ;
                ; PROCESS REGISTER ERROR
0C81 3E52       ERRR:   MVI     A,'R'    ;GET INDICATOR
0C83 210000             LXI     H,0      ;GET A 0
0C86 32B210             STA     OBUF     ;SET IN OUTPUT BUFFER
0C89 C9                 RET
                ;PROCESS SYNTAX ERROR
0C8A 3E53       ERRS:   MVI     A,'S'    ;GET INDICATOR
0C8C 32B210             STA     OBUF     ;STORE IN OUTPUT BUFFER
0C8F 210000             LXI     H,0
0C92 C35D0C             JMP     SEN1
                ;PROCESS UNDEFINED SYMBOL ERROR
0C95 3E55       ERRU:   MVI     A,'U'    ;GET INDICATOR
0C97 C38C0C             JMP     ERRS+2
                ;PROCESS VALUE ERROR
0C9A 3E56       ERRV:   MVI     A,'V'    ;GET INDICATOR
0C9C C3830C             JMP     ERRR+2
                ;PROCESS MISSING LABEL ERROR
0C9F 3E4D       ERRM:   MVI     A,'M'    ;GET INDICATOR
0CA1 32B210             STA     OBUF     ;STORE IN OUTPUT BUFFER
0CA4 CD0206             CALL    AOU1     ;DISPLAY ERROR
0CA7 C9                 RET
                ;PROCESS ARGUMENT ERROR
0CA8 3E41       ERRA:   MVI     A,'A'    ;GET INDICATOR
0CAA C38C0C             JMP     ERRS+2
                ; PROCESS OPCODE ERROR
                ; STORE 3 BYTES OF ZERO IN OBJECT CODE TO PROVIDE
                ; FOR A PATCH.
0CAD 3E4F       ERRO:   MVI     A,'O'   ;GET INDICATOR
0CAF 32B210             STA     OBUF    ;STORE IN OUTPUT BUFFER
0CB2 3A9410             LDA     PASI    ;FETCH PASS INDICATOR
0CB5 B7                 ORA     A       ;WHICH PASS
0CB6 C8                 RZ              ;RETURN IF PASS1
0CB7 0E03               MVI     C,3     ;NEED 3 BYTES
0CB9 AF         ERO1:   XRA     A       ;GET A ZERO
0CBA CDEE08             CALL    ASTO    ;PUT IN LISTING AND MEMORY
0CBD 0D                 DCR     C
0CBE C2B90C             JNZ     ERO1
0CC1 C9                 RET
                ; PROCESS LABEL ERROR
0CC2 3E4C       ERRL:   MVI     A,'L'   ;GET INDICATOR
0CC4 C3AF0C             JMP     ERRO+2
                ;PROCESS DUPLICATE LABEL ERROR
0CC7 3E44       ERRD:   MVI     A,'D'   ;GET ERROR INDICATOR
0CC9 32B210             STA     OBUF    ;STORE IN OUTPUT BUFFER
0CCC CDC506             CALL    AOUT    ;DISPLAY ERROR
0CCF C31E07             JMP     OPC     ;PROCESS OPCODE
                ;
                ;
                ; THIS ROUTINE SETS OR CLEARS BREAKPOINTS
                ;
0CD2 3A7E10     BREAK:  LDA     ABUF    ;CHECK FOR AN ARG
0CD5 B7                 ORA     A
0CD6 CA140D             JZ      CLRB    ;IF NO ARG, GO CLEAR BREAKPOINTS
0CD9 1608               MVI     D,NBR   ;ELSE, GET NUMBER OF BREAKPOINTS
0CDB 210C10             LXI     H,BRT   ;AND ADDR OF TABLE
0CDE 7E         B1:     MOV     A,M     ;GET HI BYTE OF ENTRY
0CDF 23                 INX     H.
0CE0 46                 MOV     B,M     ;GET LO BYTE OF ENTRY
0CE1 B0                 ORA     B       ;CHECK FOR EMPTY ENTRY
0CE2 CAEE0C             JZ      B2      ;BRANCH IF EMPTY
0CE5 23                 INX     H       ;ELSE GO ON TO NEXT ENTRY
0CE6 23                 INX     H
0CE7 15                 DCR     D       ;BUMP COUNT
0CE8 C2DE0C             JNZ     B1      ;AND TRY AGAIN
0CEB C35A04             JMP     WHAT    ;OOPS! NO ROOM
0CEE 2B         B2:     DCX     H
0CEF EB                 XCHG
0CF0 2A8A10             LHLD    BBUF    ;GET ADDRESS
0CF3 EB                 XCHG            ;IN D,E
0CF4 7A                 MOV     A,D     ;CHECK FOR ADDR > 11D
0CF5 B7                 ORA     A
0CF6 C2FF0C             JNZ     B3
0CF9 7B                 MOV     A,E
0CFA FE0B               CPI     11
0CFC DA5A04             JC      WHAT    ;OOPS. TOO LOW
0CFF 72         B3:     MOV     M,D     ;SAVE ADDRESS
0D00 23                 INX     H
0D01 73                 MOV     M,E
0D02 23                 INX     H
0D03 1A                 LDAX    D       ;PICK UP INSTRUCTION
0D04 77                 MOV     M,A     ;SAVE IT
0D05 3ECF               MVI     A,(RST 1) ;REPLACE IT WITH A
0D07 12                 STAX    D       ;RESTART INSTRUCTION
0D08 3EC3               MVI     A,0C3H  ;SET UP LO MEMORY
0D0A 320800             STA     8       ;WITH A JUMP TO BRKP
0D0D 212E0D             LXI     H,BRKP
0D10 220900             SHLD    9
0D13 C9                 RET             ;THEN RETURN
                ;
                ; THIS ROUTINE CLEARS ALL BREAKPOINTS
                ;
```

```
0D14 210C10     CLRB:    LXI       H,BRT     ;GET TABLE ADDRESS
0D17 0608                MVI       B,NBR     ;GET NUMBER OF BREAKPOINTS
0D19 AF         CLBL:    XRA       A         ;GET A ZERO
0D1A 56                  MOV       D,M       ;GET HI-BYTE OF ENTRY
0D1B 77                  MOV       M,A
0D1C 23                  INX       H
0D1D 5E                  MOV       E,M       ;GET LO-BYTE OF ENTRY
0D1E 77                  MOV       M,A
0D1F 23                  INX       H
0D20 46                  MOV       B,M       ;GET INST BYTE
0D21 23                  INX       H
0D22 7A                  MOV       A,D       ;WAS THIS A NULL ENTRY
0D23 B3                  ORA       E
0D24 CA290D              JZ        CL2       ;BRANCH IF IT WAS
0D27 78                  MOV       A,B
0D28 12                  STAX      D         ;ELSE, PLUG INST BACK IN
0D29 05         CL2:     DCR       B         ;BUMP COUNT
0D2A C2190D              JNZ       CLBL      ;GO DO NEXT ONE
0D2D C9                  RET                 ;RETURN WHEN DONE
                ;
                ; COME HERE WHEN WE HIT A BREAKPOINT
                ;
0D2E 220810     BRKP:    SHLD      HOLD+8    ;SAVE H,L
0D31 E1                  POP       H         ;GET PC
0D32 2B                  DCX       H         ;ADJUST IT
0D33 220A10              SHLD      HOLD+10   ;SAVE IT
0D36 F5                  PUSH      PSW       ;SAVE FLAGS
0D37 E1                  POP       H         ;GET THEM INTO HL
0D38 220010              SHLD      HOLD      ;NOW STORE THEM FOR USER
0D3B 210000              LXI       H,0
0D3E 39                  DAD       SP        ;GET STACK POINTER
0D3F 310810              LXI       SP,HOLD+8 ;SET NEW SP
0D42 E5                  PUSH      H         ;SAVE OLD SP
0D43 D5                  PUSH      D         ;SAVE D,E
0D44 C5                  PUSH      B         ;SAVE B,C
0D45 2F                  CMA                 ;COMPLEMENT ACC
0D46 D3FF                OUT       0FFH      ;DISPLAY IT IN THE LIGHTS
0D48 31B210              LXI       SP,AREA+18 ;SET SP AGAIN
0D4B 2A0A10              LHLD      HOLD+10   ;GET PC
0D4E EB                  XCHG                ;INTO D,E
0D4F 210C10              LXI       H,BRT     ;GET ADDR OF TABLE
0D52 0608                MVI       B,NBR     ;AND NUMBER OF ENTRIES
0D54 7E         BL1:     MOV       A,M       ;GET AN ENTRY FROM THE TABLE
0D55 23                  INX       H
0D56 BA                  CMP       D         ;DOES IT MATCH
0D57 C25F0D              JNZ       BL2       ;BRANCH IF NOT
0D5A 7E                  MOV       A,M       ;ELSE GET NEXT BYTE
0D5B BB                  CMP       E         ;CHECK IT
0D5C CA680D              JZ        BL3       ;IT MATCHES!
0D5F 23         BL2:     INX       H         ;BUMP AROUND THIS ENTRY
0D60 23                  INX       H
0D61 05                  DCR       B         ;BUMP COUNT
0D62 CA5A04              JZ        WHAT      ;NOT IN OUR TABLE!
0D65 C3540D              JMP       BL1
                ;
0D68 23         BL3:     INX       H
0D69 7E                  MOV       A,M       ;GET INSTR BYTE
0D6A 12                  STAX      D         ;PUT IT BACK
0D6B AF                  XRA       A         ;CLEAR ENTRY IN TABLE
0D6C 2B                  DCX       H
0D6D 77                  MOV       M,A
0D6E 2B                  DCX       H
0D6F 77                  MOV       M,A
0D70 CD0E01              CALL      CRLF      ;RESTORE THE CARRIAGE
0D73 3A0B10              LDA       HOLD+11   ;GET HI-BYTE OF PC
0D76 CD3A02              CALL      HOUT      ;TYPE IT
```

```
0D79 3A0A10             LDA     HOLD+10 ;GET LO-BYTE OF PC
0D7C CD3A02             CALL    HOUT    ;TYEP IT
0D7F 21880D             LXI     H,BMES  ;TELL USER WHAT IT IS
0D82 CD7A02             CALL    SCRN
0D85 C36700             JMP     EOR     ;GO BACK TO COMMAND LEVEL
                ;
0D88 20425245   BMES:   DB      ' BREAK',13
0D8C 41480D
                ;
                ;
                ; THIS ROUTINE PROCEEDS FROM A BREAKPOINT
                ;
0D8F 3A7E10     PROC:   LDA     ABUF    ;CHECK FOR ARG
0D92 B7                 ORA     A
0D93 CA9C0D             JZ      P1      ;JMP IF NO ARG
0D96 2A8A10             LHLD    BBUF    ;ELSE, GET ARG
0D99 220A10             SHLD    HOLD+10 ;PLUG IT INTO PC SLOT
0D9C 310010     P1:     LXI     SP,HOLD ;SET SP TO POINT AT REG'S
0D9F F1                 POP     PSW     ;RESTORE PSW
0DA0 C1                 POP     B       ;RESTORE B,C
0DA1 D1                 POP     D       ;RESTORE D,E
0DA2 E1                 POP     H       ;GET OLD SP
0DA3 F9                 SPHL            ;RESTORE IT
0DA4 2A0A10             LHLD    HOLD+10 ;GET PC
0DA7 E5                 PUSH    H       ;PUT IT ON STACK
0DA8 2A0810             LHLD    HOLD+8  ;RESTORE H,L
0DAB C9                 RET             ;AND PROCEED
                ;
                ; SYSTEM RAM
                ;
0DAC                    ORG     1000H
                ;
                ; DEFINE BREAKPOINT REGION
                ;
0008            NBR     EQU     8       ;NUMBER OF BREAKPOINTS
1000            HOLD:   DS      12      ;REGISTER HOLD AREA
100C            BRT:    DS      3*NBR   ;BREAKPOINT TABLE
                ;
                ; FILE AREA PARAMETERS
0006            MAXFIL  EQU     6       ;MAX # OF FILES
0005            NMLEN   EQU     5       ;NAME LENGTH
000D            FELEN   EQU     NMLEN+8 ;DIRECTORY ENTRY LENGTH
1024            FILE0:  DS      NMLEN
1029            BOFP:   DS      2
102B            EOFP:   DS      2
102D            MAXL:   DS      4
1031            FILTB:  DS      (MAXFIL-1)*FELEN
1072            INSP:   DS      2       ;INSERT LINE POSITION
1072            DELP    EQU     INSP    ;DELETE LINE POSITION
000D            ASCR    EQU     13      ;ASCII CARRIAGE RETURN VALUE
1074            HCON:   DS      2
1074            ADDS    EQU     HCON    ;FIND ADDRESS
1076            FBUF:   DS      NMLEN   ;FILE NAME BUFFER
107B            FREAD:  DS      2       ;FREE ADDRESS IN DIRECTORY
107D            FEF:    DS      1       ;FREE ENTRY FOUND FLAG
107D            FOCNT   EQU     FEF     ;OUTPUT COUNTER
107E            ABUF:   DS      12      ;ASCII BUFFER
108A            BBUF:   DS      4       ;BINARY BUFFER
108E            SCNT:   DS      1
108F            DCNT:   DS      1       ;DUMP ROUTINE COUNTER
000B            NCOM    EQU     11      ;NUMBER OF COMMANDS
1090            TABA:   DS      2       ;SYMBOL TABLE END ADDRESS
1092            ASPC:   DS      2       ;ASSEMBLER PROGRAM COUNTER
1094            PASI:   DS      1       ;PASS INDICATOR
1095            NCHR:   DS      1       ;LENGTH OF STRING FOR COMPARE
1096            PNTR:   DS      2       ;LINE POINTER STORAGE
```

```
1098          NOLA:   DS      1        ;NUMBER OF LABELS
1099          SIGN:   DS      1        ;SIGN STORAGE FOR SCAN
109A          OPRD:   DS      2        ;OPERAND STORAGE
109C          OPRI:   DS      1        ;OPERAND FOUND INDICATOR
109D          TEMP:   DS      1
1072          APNT    EQU     INSP     ;ASSEMBLE LINE POINTER
108E          AERR    EQU     SCNT     ;ASSEMBLER ERROR PRINT SWITCH
109E          OIND:   DS      2        ;OUTPUT ADDRESS
0005          LLAB    EQU     5        ;LENGTH OF LABELS
10A0          AREA:   DS      18
10B2          OBUF:   DS      16       ;OUTPUT BUFFER AREA
10C2                  DS      5
10C7          IBUF:   DS      83
111A          SYMT    EQU     $        ;START OF SYMBOL TABLE
              ;
              ; TELETYPE PARAMETERS
              ;
0003          TTS     EQU     3        ;TTY STATUS PORT
0002          TTI     EQU     2        ;TTY DATA IN PORT
0002          TTO     EQU     2        ;TTY DATA OUT PORT
0002          TTYDA   EQU     2        ;TTY DATA AVAILABLE BIT
0001          TTYTR   EQU     1        ;TTY XMTR READY BIT
00FF          SWCH    EQU     0FFH     ;SWITCH REGISTER
              ;
0000                  END
```

BOOTSTRAP LOADER

The IMSAI Bootstrap Loader is a system that allows the user to get a general paper tape loader into any region of RAM using only a 32-byte key-in. It requires an ASR33 teletype. To use this loader, proceed as follows:

1. Key in the basic bootstrap given below starting at location 0000.

```
3E CE D3 03 3E 17 D3 03 21 20 00 06 F8 DB 03 E6
02 CA 0D 00 DB 02 77 3C CA 08 00 23 05 C2 0D 00
```

2. Mount the bootstrap tape in the paper tape reader on the teletype so that the block of rubouts (frames with all the holes punched out) is in the reader.

3. Set the PROGRAMMED INPUT switches to the high order 8 bits of the address where the paper tape loader is to be located, e.g., to put the loader at 5C00 hex, set the PROGRAMMED INPUT switches to 5C hex. (See the warning below.)

4. Press STOP, RESET and RUN, then manually start the paper tape reader on the teletype.

If all goes well, the tape should go through the reader, stop at the end, then the loader will print an "*" on the teletype. If this is the case, refer to the IMSAI Paper Tape Loader section to use the loader.

If the loader does not type an asterisk after the tape has gone through the reader, this means the loader was not read in correctly. Proceed as follows:

1. Check the basic bootstrap key into it as correct.

2. If the key-in is correct, check the bootstrap tape for tears or distorted holes. (These may usually be fixed with cellophane tape.)

If the key-in and bootstrap tape are correct, the problem may be dirty contacts in the teletype reader. Try repeating the bootstrap procedure from the beginning.

WARNING:

1. Since the bootstrap loader resides in location 20 hex - 120, do not try to load the paper tape loader below 200 hex or it will overlay the bootstrap.

2. Be sure to locate the loader in a region where it will not be overlayed by the program it is loading. For instance, 8K BASIC occupies locations 0000-1FFF hex, so that to load 8K BASIC, the loader should be located at or above 2000 hex.

BOOTSTRAP LOADER PROGRAM LOGIC

The Bootstrap Loader is a system that allows the user to read the Paper Tape Loader into the region of RAM that begins on a 256-word boundary using a specially formatted tape.

1. Bootstrap Tape Format:

   The Bootstrap Tape consists of two sections. The first section consists of a direct core image of the second level bootstrap (described below), preceded by a block of rub-outs. In this section of the tape, each frame corresponds directly to one data byte. The second section consists of the Paper Tape Loader in standard object format.

2. Overall Logic:

   The Bootstrap Sequence Procedure is as follows:

   a. The user keys in a simple 32-byte bootstrap, starts it up, then starts the tape reader on the teletype.

   b. The basic bootstrap reads in the second level bootstrap from the first part of the bootstrap tape and starts it up.

   c. The second level bootstrap stops the tape reader then checksums itself to make sure it was loaded correctly. If not, it hangs up.

   d. If the second level bootstrap checksums correctly, it starts the tape reader and reads in the paper tape loader from the second part of the bootstrap tape and locates it in the 256-byte page specified by the PROGRAMMED INPUT switches. If it detects an error in the tape, it stops the reader and hangs up.

   e. When the Paper Tape Loader is completely loaded, it stops the paper tape reader, then starts up the Paper Tape Loader.

3. Basic Bootstrap:

   The Basic Key-In Bootstrap was designed to be as short as possible. It merely reads in characters from the tape and stores them directly into memory. Whenever it reads in a byte of FF hex, it resets its pointer and counter. This allows it to use the block of rubouts at the beginning of the tape to synchronize on.

4. Second Level Bootstrap:

The second level bootstrap is a modified version of the Paper Tape Loader. The main differences between the two are:

a. The second level bootstrap checksums itself to make sure it was loaded properly. This is done because the Basic key-in bootstrap, for reasons of brevity, does not error checking.

b. If it encounters an error, the second level bootstrap turns off the tape and hangs up.

c. If it encounters a byte of FD hex, it substitutes the contents of the PROGRAMMED INPUT switches. This is done so that the Paper Tape Loader may be located at any 256-byte page in memory. See below.

5. Relocating the Paper Tape Loader

The Paper Tape Loader that is on the second part of the bootstrap tape was assembled to begin at FD00 hex. Since there is no instruction with op-code FD hex, the only times a byte of FD hex will appear on the tape are:

a. The high byte of the address field in the paper tape record. (Note that the high byte of the address fields of all records will be FD hex.)

b. The high byte of the address in a jump instruction.

Therefore, by substituting another value (in this case, the contents of the PROGRAMMED INPUT switches) for every occurance of FD hex, we can load the Paper Tape Loader into any 256-byte page in memory.

PAPER TAPE LOADER

The IMSAI Paper Tape Loader is a program that will load tapes in the standard object format (see appendix) from the paper tape reader on an ASR33 teletype.

If the paper tape loader is read in with the bootstrap loader (see Bootstrap Loader section), it will start itself up and print an "*" on the teletype. Otherwise, it should be manually started at its beginning address.

When the loader prints an "*" on the teletype, mount the tape to be loaded in the paper tape reader on the teletype. Then, strike any key on the teletype. The paper tape reader should start automatically. While the tape is being read in, the data being loaded will be displayed in the PROGRAMMED OUTPUT lights.

The loader will stop the reader and print an "*" under two conditions:

1. If the PROGRAMMED OUTPUT displays 00 (all lights off), the loader has encountered an End-of-File record, an the program has been successfully loaded. At this point, another tape may be loaded by placing it in the paper tape reader and striking a key on the teletype.

2. If something other than 00 is displayed in the PROGRAMMED OUTPUT lights, a bad record has been encountered in the tape. The record may be re-read as follows:

   - Move the switch on the reader to the "FREE" position

   - Back the tape up about two feet

   - Put the switch back in the "STOP" position

   - Strike a key on the teletype

   If the loader stops again on the same record, inspect the tape for tears or distorted holes (these may usually be fixed with cellophane tape).

PAPER TAPE LOADER PROGRAM LOGIC

The IMSAI Paper Tape Loader is a program designed to load paper tapes in the standard object format from the paper tape reader on an ASR33 teletype. The loader is designed to use no stack or local RAM, thereby allowing it to be executed out of ROM.

1. Object Tape Format:

The standard object format is a blocked hexadecimal format. The data on the tape is blocked into discrete records, each record containing record length, record type, memory address and checksum information in addition to data. A frame-by-frame description is as follows:

Frame 0	<u>Record Mark</u>. Signals the start of a record. The ASCII character colon (":" 3A hex) is used as the record mark.
Frames 1,2 (0-9, A-F)	<u>Record Length</u>. Two ASCII characters representing a hexadecimal number in the range 0 to FF (0 to 255). This is the count of actual data bytes in the record type or checksum. A record length of 0 indicates end-of-file.
Frames 3 to 6	<u>Load Address</u>. Four ASCII characters that represent the initial memory location where the data following will be loaded. The first data byte is stored in the location pointed to by the load address; succeeding data bytes are loaded into ascending addresses.
Frames 7,8	<u>Record Type</u>. Two ASCII characters. Currently all records are type 0. This field is reserved for future expansion.
Frames 9 to 9+2*	<u>Data</u>. Each 8-bit memory word is represented by two frames containing the ASCII characters 0-9, A-F) to represent a hexadecimal value 0 to FF hex (0 to 255).
Frames 9+2* (Record Length) to 9+2* (Record Length + 1	<u>Checksum</u>. The checksum is the negative of the sum of all 8-bit bytes in the record since the record mark (":") evaluated modulus 256. That is, if you add together all the 8-bit bytes, ignoring all carries out of an 8-bit sum then add the checksum, the result is zero.

Example: If memory locations 1 through 3 contain 53F8EC, the format of the hex file produced when these locations are punched is:

```
:0300010053F8ECC5
```

2. Register Allocation:

Since this loader uses no RAM, all variables and data are kept in the registers. The registers are assigned as follows:

A	- scratch
B	- byte count for data field
C	- checksum
D	- holes the data byte
E	- flag register, describes what to do next

If this register contains zero, this program is looking for a ":" to signal the beginning of a block. Otherwise, if bit 7=1, then the next character is the first digit of a byte. If bit 2=0, the next character is the second digit of a byte. Bits 0-6 have the following significance:

1	- next byte is a count
2	- next byte is a high byte of the load address
3	- next byte is a low byte of the load address
4	- next byte is a type byte
5	- next byte is a data byte
6	- next byte is a checksum byte.

H, L - Load Address.

3. Logic:

The program flow is controlled by the flags in the E-register as given above.

```
                    ;
                    ; *** BASIC KEY-IN BOOTSTRAP LOADER ***
                    ;
                    ; THIS SIMPLE LOADER BOOTSTRAPS IN THE SECOND
                    ; LEVEL BOOTSTRAP, WHICH IN TURN LOADS THE
                    ; REAL PAPER TAPE LOADER.
                    ;
                    ; TO USE THIS LOADER, PROCEED AS FOLLOWS:
                    ;       (1) KEY IN THIS LOADER, STARTING AT LOC 1000
                    ;       (2) MOUNT THE BOOTSTRAP TAPE, SO THAT
                    ;           THE BLOCK OF RUBOUTS AT THE BEGINNING
                    ;           OF THE TAPE IS IN THE READER
                    ;       (3) SET THE PROGRAMMED INPUT SWITCHES TO THE
                    ;           HIGH ORDER 8 BITS OF THE ADDRESS WHERE
                    ;           YOU WANT THE PAPER TAPE LOADER TO
                    ;           BE LOADED. (E.G. TO CAUSE THE LOADER
                    ;           TO BE LOADED AT 5C00, SET THE PROGRAMMED
                    ;           INPUT SWITCHES TO 5C.)
                    ;       (4) PRESS THE 'RESET' KEY, FOLLOWED BY THE
                    ;           'RUN' KEY, THEN MANUALLY START THE PAPER
                    ;           TAPE READER ON THE TELETYPE.
                    ;
                    ;  IF EVERYTHING GOES CORRECTLY, THE LOADER WILL STOP
                    ;  THE PAPER TAPE READER, AND PRINT A * ON THE
                    ;  TELETYPE.  AT THIS POINT, MOUNT THE TAPE TO BE
                    ;  LOADED IN THE TELETYPE READER, THEN STRIKE ANY KEY
                    ;  ON THE TELETYPE.  THE LOADER WILL START THE
                    ;  PAPER TAPE READER, AND START LOADING THE TAPE. IF
                    ;  IT FINDS ANYTHING WRONG WITH THE TAPE, IT WILL
                    ;  STOP THE READER.  LOADING MAY BE CONTINUED BY
                    ;  STRIKING A KEY ON THE TELETYPE.
                    ;
                    ;
00F8                CNT     EQU     0F8H    ;SIZE OF 2ND LEVEL BOOTSTRAP
                    ;
0000 3ECE           BOOT1:  MVI     A,0CEH  ;GET MODE BYTE FOR SIO BOARD.
0002 D303                   OUT     03      ;ISSUE IT
0004 3E17                   MVI     A,17H   ;GET COMMAND BYTE
0006 D303                   OUT     03      ; ISSUE IT
0008 212000         B1RST:  LXI     H,B1END ;GET LOAD ADDRESS
000B 06F8                   MVI     B,CNT   ;GET # OF BYTES
                    ;
000D DB03           LOOP:   IN      03      ;GET STATUS
000F E602                   ANI     2       ;IS THERE A BYTE READY
0011 CA0D00                 JZ      LOOP    ;KEEP WAITING
0014 DB02                   IN      2       ;GET THE BYTE
0016 77                     MOV     M,A     ;STORE IT
0017 3C                     INR     A       ;WAS IT A RUBOUT?
0018 CA0800                 JZ      B1RST   ;IF YES, RESET POINTERS
001B 23                     INX     H       ;ELSE, BUMP POINTER
001C 05                     DCR     B       ;AND DECR COUNT
001D C20D00                 JNZ     LOOP    ;IF NOT DONE, GO GET ANOTHER
                                            ; CHAR. ELSE, FALL THROUGH AND
                                            ; START UP SECOND LEVEL
                                            ; BOOTSTRAP.
0020                B1END   EQU     $
0000                        END
```

```
                  ;
                  ; SECOND LEVEL BOOTSTRAP
                  ;
                  ; THIS LOADER IS PULLED IN BY THE BASIC KEY-IN
                  ; LOADER. WHEN STARTED UP BY THE KEY-IN LOADER,
                  ; IT CHECKSUMS ITSELF, TO MAKE SURE THAT IT
                  ; HAS BEEN LOADED CORRECTLY, THEN PULLS IN AND
                  ; RELOCATES THE MAIN PAPERTAPE LOADER.
                  ;
                  ; NOTE THAT THIS LOADER IS A SLIGHTLY MODIFIED
                  ; VERSION OF THE MAIN PAPER TAPE LOADER.
                  ;
                  ;
0000                      ORG     20H
                  ;
0020 3E13         BOOT2:  MVI     A,13H   ;GET STOP CHAR
0022 D302                 OUT     2       ;STOP THE READER
0024 06F7                 MVI     B,CHKSM-BOOT2 ;GET SIZE OF LDR
0026 212000               LXI     H,BOOT2 ;GET ADDRESS OF LDR
0029 AF                   XRA     A       ;CLEAR A AND CARRY
                  ;
                  ; PERFORM AN END-AROUND CHECKSUM, TO MAKE SURE
                  ; WE WERE LOADED CORRECTLY
                  ;
002A 8E           CHECK:  ADC     M       ;ADD IN A BYTE WITH CARRY
002B 23                   INX     H       ;BUMP POINTER
002C 05                   DCR     B       ;DECREMENT COUNT
002D C22A00               JNZ     CHECK   ;KEEP GOING
0030 CE00                 ACI     0       ;ADD IN LAST CARRY
0032 BE                   CMP     M       ;COMPARE WITH CHECKSUM
0033 C23300       XXX:    JNZ     XXX     ;HANG UP IF NO GOOD.
                  ;
                  ; WE DO THE FOLLOWING NONSENSE BECAUSE THE
                  ; BASIC KEY-IN BOOTSTRAP WILL NOT LOAD
                  ; AN OFFH CHARACTER.
                  ;
0036 21BC00               LXI     H,FF1+1 ;GET ADDRESS OF 'IN 0FEH' INST
0039 34                   INR     M       ;MAKE IT 'IN 0FFH'.
003A 21B100               LXI     H,FF2+1 ;DO IT AGAIN
003D 34                   INR     M
003E 210B01               LXI     H,FF3+1 ;AND AGAIN
0041 34                   INR     M
                  ;
                  ; NOW WE'RE READY TO LOAD AND RELOCATE THE LOADER
                  ;
0042 C35E00               JMP     STR     ;1ST TIME, SKIP RE-INIT STUFF.
                  ;
0045 3EAA         START:  MVI     A,0AAH  ;GET DUMMY MODE BYTE
0047 D303                 OUT     3
0049 3E40                 MVI     A,40H   ;GET RESET COMMAND
004B D303                 OUT     3       ;ISSUE IT
004D 3EFA                 MVI     A,0FAH
004F D303                 OUT     3       ;ISSUE MODE BYTE TO SIO
0051 3E17                 MVI     A,17H
0053 D303                 OUT     3       ;ISSUE COMMAND BYTE
0055 DB03         SL:     IN      03      ;GET STATUS
0057 E602                 ANI     02      ;CHECK FOR CHAR READY
0059 CA5500               JZ      SL      ;KEEP WAITING
005C DB02                 IN      02      ;READ CHAR AND IGNOR
005E DB03         STR:    IN      03      ;GET STATUS
0060 E601                 ANI     1       ;MAKE SURE WE HAVE XMTR RDY
0062 CA5E00               JZ      STR
0065 3E11                 MVI     A,11H   ;GET 'XON' CHAR
0067 D302                 OUT     02      ;START READER
```

```
                  ;
0069 1E00         LOOP1:   MVI     E,0       ;CLEAR FLAG
006B 0E00                  MVI     C,0       ;CLEAR CHECKSUM
                  ;
006D DB03         LOOP2:   IN      3         ;GET SIO STATUS
006F E602                  ANI     2         ;CHECK FOR CHARACTER
0071 CA6D00                JZ      LOOP2     ;KEEP WAITING
0074 7B                    MOV     A,E       ;GET FLAG
0075 B7                    ORA     A         ;IS IT ZERO?
0076 C28700                JNZ     X1        ;NO, GO PROCESS A HEX CHAR
0079 DB02                  IN      2         ;YES, WE'RE LOOKING FOR A COLON
007B E67F                  ANI     127       ;STRIP OFF PARITY BIT
007D FE3A                  CPI     ':'       ;IS IT A COLON?
007F C26D00                JNZ     LOOP2     ;NO, KEEP WAITING
0082 1E81                  MVI     E,81H     ;YES, SET FLAG FOR COUNT BYTE
0084 C36D00                JMP     LOOP2     ;AND GET ANOTHER CHAR.
                  ;
                  ; WE'RE PUTTING TOGETHER A BYTE. FLAG BIT 7 = 1 => HIGH
                  ; DIGIT OF BYTE, BIT 7=0 => LOW DIGIT
                  ;
0087 F2A200       X1:      JP      Y1        ;JUMP IF LOW DIGIT
008A E67F                  ANI     127       ;ELSE STRIP OFF HIGH BIT
008C 5F                    MOV     E,A       ;PUT FLAG BACK IN E-REG
008D DB02                  IN      2         ;GET THE CHAR
008F E67F                  ANI     127       ;STRIP OFF THE PARITY BIT
0091 FE3A                  CPI     '9'+1     ;IS IT .LE. '9'
0093 FA9800                JM      X2        ;SKIP IT YES
0096 C609                  ADI     9         ;IF NOT, ADJUST IT
0098 E60F         X2:      ANI     0FH       ;GET HEX DIGIT
009A 87                    ADD     A         ;SHIFT LEFT ONE BIT
009B 87                    ADD     A         ;     TWO BITS
009C 87                    ADD     A         ;     THREE BITS
009D 87                    ADD     A         ;AND FOUR BITS.
009E 57                    MOV     D,A       ;SAVE NIBBLE IN D REG
009F C36D00                JMP     LOOP2
                  ;
                  ; PROCESS LOW DIGIT OF BYTE, THEN DECIDE WHAT TO DO WITH
                  ;
00A2 DB02         Y1:      IN      2         ;GET THE CHAR
00A4 E67F                  ANI     127       ;GET RID OF PARITY BIT
00A6 FE3A                  CPI     '9'+1     ;HEX IS SUCH A PAIN.
00A8 FAAD00                JM      Y2
00AB C609                  ADI     9
00AD E60F         Y2:      ANI     0FH
00AF B2                    ORA     D         ;MAKE THE BYTE
00B0 D3FE         FF2:     OUT     0FEH      ;PUT IT IN LIGHTS
00B2 57                    MOV     D,A       ;SAVE IT IN D REG
00B3 81                    ADD     C         ;ADD IT INTO CHECKSUM
00B4 4F                    MOV     C,A       ;SAVE RUNNING CHECKSUM
00B5 7A                    MOV     A,D       ;GET BYTE BACK
00B6 FEFD                  CPI     0FDH      ;IS IT FELOCATABLE BYTE?
00B8 C2BD00                JNZ     Y3        ;BRANCH IF NOT
00BB DBFE         FF1:     IN      0FEH      ;ELSE SUBSTITUE SWITCHS
00BD 57           Y3:      MOV     D,A       ;PUT BYTE BACK IN D
00BE 7B                    MOV     A,E       ;GET FLAG IN A
00BF 3D                    DCR     A         ;THEN DISPATCH  ON IT
00C0 CA0401                JZ      COUNT
00C3 3D                    DCR     A
00C4 CAFE00                JZ      HADD
00C7 3D                    DCR     A
00C8 CAF800                JZ      LADD
00CB 3D                    DCR     A
00CC CAF300                JZ      TYPE
00CF 3D                    DCR     A
00D0 CAE700                JZ      PUT
00D3 79                    MOV     A,C       ;MUST BE TIME TO CHECK THE
```

```
00D4 B7                    ORA     A         ; CHECKSUM. IS IT ZERO?
00D5 CA6900                JZ      LOOP1     ;YES, GO GET NEXT RECORD
00D8 214500                LXI     H,START   ;ELSE, GET RESTART ADDR
00DB 3E13          STOP:   MVI     A,13H     ;GET 'XOFF' CHAR
00DD D302                  OUT     2         ;TURN OFF READER
00DF DB03          STPL:   IN      3         ;WAIT TILL XMTR BUFFER EMPTY
00E1 E604                  ANI     4
00E3 CADF00                JZ      STPL
00E6 E9                    PCHL              ;GO AWAY.
                   ;
                   ;
                   ; PUT A DATA BYTE INTO CORE
                   ;
00E7 72            PUT:    MOV     M,D       ;STORE THE DATA
00E8 23                    INX     H         ;INCREMENT THE H REG
00E9 1E85                  MVI     E,85H     ;RESET FLAG FOR NEXT DATA BYTE
00EB 05                    DCR     B         ;DECR COUNT
00EC C26D00                JNZ     LOOP2     ;GO BACK FOR MORE DATA.
00EF 1C                    INR     E         ;OUT OF DATA, SET FLAG FOR
00F0 C36D00                JMP     LOOP2     ;  CHECKSUM.
                   ;
                   ; IGNORE A TYPE BYTE
                   ;
00F3 1E85          TYPE:   MVI     E,85H     ;SET FLAG FOR DATA
00F5 C36D00                JMP     LOOP2     ;GO GET DATA
                   ;
                   ; GET LOW BYTE OF ADDRESS
                   ;
00F8 6A            LADD:   MOV     L,D       ;GET BYTE INTO L-REG
00F9 1E84                  MVI     E,84H     ;SET FLAG FOR TYPE BYTE
00FB C36D00                JMP     LOOP2
                   ;
                   ; GET HIGH BYTE OF ADDRESS
                   ;
00FE 62            HADD:   MOV     H,D       ;GET BYTE INTO H
00FF 1E83                  MVI     E,83H     ;SET FLAG FOR LOW ADDRESS BYTE
0101 C36D00                JMP     LOOP2
                   ;
                   ; GET COUNT BYTE
                   ;
0104 42            COUNT:  MOV     B,D       ;PUT COUNT INTO B
0105 7A                    MOV     A,D       ;CHECK FOR EOF
0106 B7                    ORA     A
0107 C21201                JNZ     C1        ;IF NOT EOF, CONTINUE
010A DBFE          FF3:    IN      0FEH      ;GET HIGH BYTE OF LOADER
010C 67                    MOV     H,A       ; ADDRESS INTO H
010D 2E00                  MVI     L,0       ;AND LOW BYTE
010F C3DB00                JMP     STOP      ;STOP TAPE, THEN GOTO LOADER.
                   ;
0112 1E82          C1:     MVI     E,82H     ;SET FLAG FOR ADDRESS BYTE
0114 C36D00                JMP     LOOP2
                   ;
                   ;
0117 C8            CHKSM:  DB      0C8H      ;SELF-CHECKSUM FOR THIS LOADER
                   ;
0000                       END
```

```
                    ;
                    ; *** IMSAI PAPER TAPE LOADER ***
                    ;
                    ;       REV 0   3/3/76
                    ;
                    ; THIS LOADER IS DESIGNED TO LOAD PAPER TAPES IN
                    ; THE STANDARD OBJECT FORMAT (SEE THE SOFTWARE
                    ; SECTION OF THE 8080 USER MANUAL) FROM AN ASR 33
                    ; TELETYPE.  IT USES NO STACK AND NO LOCAL RAM, SO
                    ; THAT IT MAY BE RUN FROM PROM WITHOUT REQUIRING
                    ; A RAM CARD OF ITS OWN.
                    ;
                    ; USING THE LOADER:
                    ;       IF THIS LOADER IS BROUGHT IN WITH THE
                    ; BOOTSTRAP SEQUENCE (DOCUMENTED ELSEWHERE),
                    ; IT WILL START ITSELF UP. OTHERWISE, MANUALLY
                    ; START IT AT ITS BEGINNING.  IT WILL RESPOND
                    ; BY TYPING A * ON THE TELETYPE. MOUNT THE TAPE
                    ; TO BE LOADED IN THE READER, AND STRIKE ANY KEY.
                    ; THE LOADER WILL START THE READER AUTOMATICALLY.
                    ;       THE LOADER WILL STOP THE TAPE AND TYPE A * IN
                    ; EITHER OF TWO CASES:
                    ;
                    ;       (1) IT HAS SEEN AN END OF FILE RECORD. IN
                    ;           THIS CASE, ZERO WILL BE DISPLAYED IN
                    ;           THE PROGRAMMED OUTPUT LIGHTS.
                    ;
                    ;       (2) IT ENCOUNTERED A BAD RECORD. IN THIS CASE
                    ;           AN NON-ZERO QUANTITY WILL BE DISPLAYED
                    ;           IN THE PROGRAMMED OUTPUT LIGHTS.
                    ;
                    ; IN EITHER CASE, LOADING MAY BE CONTINUED BY STRIKING
                    ; A KEY.
                    ;
                    ;
0000                        ORG     0FD00H
                    ;
FD00 110100         START:  LXI     D,1     ;WAIT ABOUT A SECOND SO A
FD03 210000                 LXI     H,0     ;  PREVIOUS 'XOFF' CHARACTER
FD06 19             SL0:    DAD     D       ;  HAS TIME TO STOP THE READER
FD07 D206FD                 JNC     SL0
                    ;
                    ; INITIALIZE SIO BOARD.
                    ;
FD0A 3EAA                   MVI     A,0AAH  ;GET DUMMY MODE BYTE
FD0C D303                   OUT     3
FD0E 3E40                   MVI     A,40H   ;GET RESET COMMAND
FD10 D303                   OUT     3       ;ISSUE IT
FD12 3EFA                   MVI     A,0FAH
FD14 D303                   OUT     3       ;ISSUE MODE BYTE TO SIO
FD16 3E17                   MVI     A,17H
FD18 D303                   OUT     3       ;ISSUE COMMAND BYTE
FD1A 3E2A                   MVI     A,'*'   ;GET AN ASTERISK
FD1C D302                   OUT     02      ;PRINT IT
FD1E DB02                   IN      02      ;THROW AWAY ANY CHAR IN BUFFER
FD20 DB03           SL2:    IN      03      ;GET STATUS
FD22 E602                   ANI     02      ;CHECK FOR CHAR READY
FD24 CA20FD                 JZ      SL2     ;KEEP WAITING
FD27 DB02                   IN      02      ;READ CHAR AND IGNOR
FD29 3E11                   MVI     A,11H   ;GET 'XON' CHAR
FD2B D302                   OUT     02      ;START READER
                    ;
FD2D 1E00           LOOP1:  MVI     E,0     ;CLEAR FLAG
FD2F 0E00                   MVI     C,0     ;CLEAR CHECKSUM
```

```
                  ;
FD31 DB03       LOOP2:  IN      3          ;GET SIO STATUS
FD33 E602               ANI     2          ;CHECK FOR CHARACTER
FD35 CA31FD             JZ      LOOP2      ;KEEP WAITING
FD38 7B                 MOV     A,E        ;GET FLAG
FD39 B7                 ORA     A          ;IS IT ZERO?
FD3A C24BFD             JNZ     X1         ;NO, GO PROCESS A HEX CHAR
FD3D DB02               IN      2          ;YES, WE'RE LOOKING FOR A COLON
FD3F E67F               ANI     127        ;STRIP OFF PARITY BIT
FD41 FE3A               CPI     ':'        ;IS IT A COLON?
FD43 C231FD             JNZ     LOOP2      ;NO, KEEP WAITING
FD46 1E81               MVI     E,81H      ;YES, SET FLAG FOR COUNT BYTE
FD48 C331FD             JMP     LOOP2      ;AND GET ANOTHER CHAR.
                  ;
                  ; WE'RE PUTTING TOGETHER A BYTE. FLAG BIT 7 = 1 => HIGH
                  ; DIGIT OF BYTE, BIT 7=0 => LOW DIGIT
                  ;
FD4B F266FD     X1:     JP      Y1         ;JUMP IF LOW DIGIT
FD4E E67F               ANI     127        ;ELSE STRIP OFF HIGH BIT
FD50 5F                 MOV     E,A        ;PUT FLAG BACK IN E-REG
FD51 DB02               IN      2          ;GET THE CHAR
FD53 E67F               ANI     127        ;STRIP OFF THE PARITY BIT
FD55 FE3A               CPI     '9'+1      ;IS IT .LE. '9'
FD57 FA5CFD             JM      X2         ;SKIP IT YES
FD5A C609               ADI     9          ;IF NOT, ADJUST IT
FD5C E60F       X2:     ANI     0FH        ;GET HEX DIGIT
FD5E 87                 ADD     A          ;SHIFT LEFT ONE BIT
FD5F 87                 ADD     A          ;    TWO BITS
FD60 87                 ADD     A          ;    THREE BITS
FD61 87                 ADD     A          ;AND FOUR BITS.
FD62 57                 MOV     D,A        ;SAVE NIBBLE IN D REG
FD63 C331FD             JMP     LOOP2
                  ;
                  ; PROCESS LOW DIGIT OF BYTE, THEN DECIDE WHAT TO DO WITH
                  ;
FD66 DB02       Y1:     IN      2          ;GET THE CHAR
FD68 E67F               ANI     127        ;GET RID OF PARITY BIT
FD6A FE3A               CPI     '9'+1      ;HEX IS SUCH A PAIN.
FD6C FA71FD             JM      Y2
FD6F C609               ADI     9
FD71 E60F       Y2:     ANI     0FH
FD73 B2                 ORA     D          ;MAKE THE BYTE
FD74 D3FF               OUT     0FFH       ;PUT IT IN LIGHTS
FD76 57                 MOV     D,A        ;SAVE IT IN D REG
FD77 81                 ADD     C          ;ADD IT INTO CHECKSUM
FD78 4F                 MOV     C,A        ;SAVE RUNNING CHECKSUM
FD79 7B                 MOV     A,E        ;GET FLAG IN A
FD7A 3D                 DCR     A          ;THEN DISPATCH  ON IT
FD7B CAC1FD             JZ      COUNT
FD7E 3D                 DCR     A
FD7F CABBFD             JZ      HADD
FD82 3D                 DCR     A
FD83 CAB5FD             JZ      LADD
FD86 3D                 DCR     A
FD87 CAB0FD             JZ      TYPE
FD8A 3D                 DCR     A
FD8B CAA4FD             JZ      PUT
FD8E 79                 MOV     A,C        ;MUST BE TIME TO CHECK THE
FD8F B7                 ORA     A          ; CHECKSUM. IS IT ZERO?
FD90 CA2DFD             JZ      LOOP1      ;YES, GO GET NEXT RECORD
FD93 2F         STOP:   CMA                ;DISPLAY REASON FOR STOPPING
FD94 D3FF               OUT     0FFH
FD96 3E13               MVI     A,13H      ;ELSE, GET 'XOFF' CHAR
FD98 D302               OUT     2          ;TURN OFF READER
FD9A DB03       STPL:   IN      3          ;WAIT TILL XMTR BUFFER EMPTY
FD9C E604               ANI     4
```

```
        FD9E CA9AFD             JZ      STPL
        FDA1 C300FD             JMP     START
                        ;
                        ;
                        ; PUT A DATA BYTE INTO CORE
                        ;
        FDA4 72         PUT:    MOV     M,D     ;STORE THE DATA
        FDA5 23                 INX     H       ;INCREMENT THE H REG
        FDA6 1E85               MVI     E,85H   ;RESET FLAG FOR NEXT DATA BYTE
        FDA8 05                 DCR     B       ;DECR COUNT
        FDA9 C231FD             JNZ     LOOP2   ;GO BACK FOR MORE DATA.
        FDAC 1C                 INR     E       ;OUT OF DATA, SET FLAG FOR
        FDAD C331FD             JMP     LOOP2   ;  CHECKSUM.
                        ;
                        ; IGNORE A TYPE BYTE
                        ;
        FDB0 1E85       TYPE:   MVI     E,85H   ;SET FLAG FOR DATA
        FDB2 C331FD             JMP     LOOP2   ;GO GET DATA
                        ;
                        ; GET LOW BYTE OF ADDRESS
                        ;
        FDB5 6A         LADD:   MOV     L,D     ;GET BYTE INTO L-REG
        FDB6 1E84               MVI     E,84H   ;SET FLAG FOR TYPE BYTE
        FDB8 C331FD             JMP     LOOP2
                        ;
                        ; GET HIGH BYTE OF ADDRESS
                        ;
        FDBB 62         HADD:   MOV     H,D     ;GET BYTE INTO H
        FDBC 1E83               MVI     E,83H   ;SET FLAG FOR LOW ADDRESS BYTE
        FDBE C331FD             JMP     LOOP2
                        ;
                        ; GET COUNT BYTE
                        ;
        FDC1 42         COUNT:  MOV     B,D     ;PUT COUNT INTO B
        FDC2 7A                 MOV     A,D     ;CHECK FOR EOF
        FDC3 B7                 ORA     A
        FDC4 CA93FD             JZ      STOP    ;IF EOF, GO STOP READER
        FDC7 1E82               MVI     E,82H   ;ELSE SET FLAG FOR ADDRESS BYTE
        FDC9 C331FD             JMP     LOOP2
                        ;
                        ;
        0000                    END
```