8080/8085 SIMULATOR MANUAL

Microtec
P.O. Box 60337
Sunnyvale, CA 94088
408-733-2919

TABLE OF CONTENTS

INTRODUCTION

Microtec has developed an Interactive Simulator for the 8080/8085 microprocessor. The Simulator program is written in FORTRAN IV to achieve compatibility with most computer systems. The program is approximately 4200 FORTRAN statements in length, twenty percent of which are comments. The program is written in ANSI standard FORTRAN IV and no facility peculiar to any one computer was utilized. This was done in order to eliminate FORTRAN compatibility problems. The program is modular and may be executed in an overlay mode should memory restrictions make that necessary.

Although the Simulator is most effective in an interactive environment, it may also be used in a batch mode.

The program simulates all aspects of the microprocessor, including interrupts. The full 64K byte microprocessor memory is simulated. The Simulator provides for the unlimited setting of instruction breakpoints and the ability to trace or single step the execution of each instruction. A very extensive I/O capability is provided that allows the user to perform I/O simulation interactively or from files and Data Tables. Symbolic debugging is also provided; the Simulator can read in symbol values from the assembler, and the user may then use these symbols as command arguments instead of absolute addresses.

This manual provides the information pertinent to the operation and use of the Simulator, but it does not describe programming techniques or the operation of the 8080/8085 microprocessor. The reader may consult the manufacturer's literature for this information.

THE SIMULATOR

Overview

This program simulates the operation of the 8080/8085 microprocessor by implementing, in software, the registers and logic control functions of the actual microprocessor. Like the actual microprocessor, the Simulator's simulated memory must be loaded with an object module which contains the program to be debugged. The Load or LS command is used to do this. The object program may be generated through use of any appropriate Assembler Program, but it must be in standard Intel Hexadecimal format (see Appendix C). Microtec's MASM80 Assembler will perform this function.

After the object module has been loaded, the user may use the Simulator commands to initialize the various registers and status bits. The RES command may be used to simulate an actual microprocessor Reset, or the simulated Program Counter may be set to a specific address to debug a particular subroutine. Simulation is initiated through use of the Execute or Trace commands. When program execution begins, the Simulator fetches the instruction pointed to by the Program Counter from the simulated memory and executes it. Memory Registers and Status Bits are then changed to reflect the results of the instruction execution.

The Simulator offers several advantages over other methods of program debugging. One of the most obvious ones is that program debugging may be performed before the hardware is actually built, or when it is not available because someone else is using it to debug his program. Another advantage of the Simulator is that program execution can be controlled precisely. Microprocessor registers and flags, not normally accessable, may be examined and modified. By using appropriate commands, the user may trace program flow, examine and modify memory locations, feed test data to input ports, view output data, and as a result, determine the correct operation of the program under test.

Another useful feature implemented in the Simulator is symbolic debugging. Symbols (labels) from the Assembly program may be read into the Simulator. The symbols may then be used as command arguments when performing functions such as setting breakpoints. This reduces the need to refer to absolute addresses, especially helpful when debugging code written in a higher level language.

The program can be executed in a batch or interactive mode. Provision is made for extensive Input/Output capability with the host computer. Commands may be read from disk files or any logical device (card reader) as well as from the controlling terminal. Simulator output may be written to a disk file or any logical device (line printer) as well as to the controlling terminal.

Processor Model

As previously mentioned, the Simulator has internal variables and arrays that correspond to all of the microprocessor registers and status bits. These elements may be initialized through use of the Simulator's SET Command.

The full 64K word microprocessor memory is simulated and kept on a disk file. However, segments or pages of the memory that are being accessed are kept in an array in main memory. By only keeping the pages being accessed in main memory, the size of the simulation program on the host computer is kept down to a reasonable limit. A multi-page scheme keeps disk page swapping to a minimum, allowing rapid Simulator execution.

Memory may be made to have the characteristics of Read Only Memory (ROM) through use of the Protect Command. Protected memory may not be written to. The Protect Command description discusses this feature further.

There is also an Input/Output memory inside the Simulator that holds the last value read from an Input Port and the last value written to an Output Port. The output values are placed in this array as well as written to the file or logical device specified by the Output Port Commands. Likewise, data values written to and read from memory mapped I/O ports are saved in the corresponding memory locations as well as written to the file or logical device specified. The last values read from and written to the serial port are saved and may be examined with the DIM command.

Inputs on the Reset and Interrupt Pins may be simulated through use of the Reset and Interrupt Commands. Further details are given in the command explanations.

The Simulator keeps a cumulative cycle count of the total number of microprocessor cycles executed. This count may be used to calculate routine execution times. The SET command may be used to initialize this count.

Simulation Modes

The following terms describe those conditions or modes in which the Simulator operates and which are referred to throughout this manual. Modes are not exclusive. The Simulator may be in the Command Mode and the Read Mode.

Command Mode - the program is considered to be in the command mode whenever it is requesting and executing user input commands. It leaves the command mode only when a command is recognized that requires instruction execution.

Execute Mode - this mode is entered from the command mode. Any of the following commands cause the Simulator to enter the execute mode: T, TA, TB, E, EA. When simulated program execution is complete, the command mode is returned to. Also, any errors that occur during the execution mode will cause the program to return to the command mode.

Read Mode - the Simulator enters the read mode with respect to I/O input assignments whenever the user specifies a R, RD, RE, or RED command to change the current command input device. Note that the program is said to be in the read mode even if the device assignment specified in one of the read commands is the standard input device.

Write Mode - the Simulator enters the write mode with respect to I/O output assignments whenever the user specifies the W command. This is the case even if the device assignment specified in the command is the standard output device.

Standard I/O refers to those I/O device assignments which were defined in the program at compile time. They are the I/O device assignments that will be used when the program is first executed.

Program Operation

When the Simulator is executed, a header is printed on the standard output device (usually a terminal) indicating that the Simulator has been entered. Commands are initially read from this device but may be read from other devices or files through use of the "R" and "RE" commands. When a command is entered, it is checked for validity and then executed. Any output (trace information, I/O messages, etc.) is written to the standard output device or to an alternate I/O device or file if specified by a "W" command.

If an error occurs during command interpretation or execution, an appropriate error message will be written to the current output device and to the standard output device (usually a CRT terminal in the interactive mode). The current output device could be a line printer. An error will also cause all I/O device assignments to be returned to the "standard I/O devices." This means that if an error occurs after an "R" or "RE" command has been specified causing commands to be read from an alternate device, the program will return to reading commands from the standard input device.

The program may be installed to run in an interactive or batch mode. The differences are:

1. In the interactive mode a prompt character will be displayed to request each command. This feature may be eliminated by the user if the host system also displays a prompt. If the user has specified a "R" or "RE" command, changing the command input device, the prompt will not be displayed in the interactive mode. No prompt character is displayed in the batch mode.

2. Command or execution errors in the interactive mode cause all standard I/O device assignments to become active and cause the Simulator to prompt for the next command from the

standard interactive input device. Command errors in the batch mode are fatal.

3. If an end-of-file (EOF) is detected during a "R" or "RE" command, the Simulator will return to the standard input device to read commands in both batch and interactive modes. If an EOF is detected while reading commands from the standard input device, the program will terminate in the batch mode. In the interactive mode, the Simulator will execute only one instruction. A blank line is treated the same as detecting an EOF in both the batch and interactive modes. This allows users who cannot detect an EOF to enter a space followed by a carriage return to simulate an EOF condition.

4. During the simulation of an input instruction, the program will sometimes display the message "*INPUT PORT N =" to request the input value from the user. This will only be done in the interactive mode when not in read mode. In the batch mode the program will read the input value from the specified I/O device without displaying any message.

5. In the batch mode, commands read from the standard input device will be echoed to the current output device. In the interactive mode, commands will not be echoed unless specified by the "RE" command.

System Input/Output

There are several commands in the program which utilize the I/O capabilities of the host computer. Object modules and Simulator commands are read from the host computer's logical devices or disk files. Simulator output may be written to any logical device or disk file.

At compile time, a set of I/O devices called the "standard I/O devices" are defined. These are defined for Command input, Object Module input, and Simulator output. These are the devices used by the Simulator when it is first executed. Through use of the Read and Write Commands, I/O may be performed with disk files and logical devices different from the standard devices. The following three paragraphs summarize the types of I/O the Simulator can perform:

1. I/O is performed with the standard I/O devices defined at compile time. This is the standard method of performing I/O and is in effect if no Read or Write commands are specified.

2. I/O is performed with an alternate I/O device such as a card reader or line printer. To do this the user must specify the appropriate logical device number as the Read or Write command argument. The unit number specified may even be one of the standard I/O devices.

3. I/O is performed with a system file. To do this the user must specify the file name as the Read or Write Command argument. File names must begin with an alphabetic character, not a number. The Simulator will open the file and perform the required I/O.

The Input device or file currently being used to read Simulator commands is called the "Current Input Device." Likewise, the output device that the Simulation listing is being written to is called the "Current Output Device."

Addressing

Many of the simulator commands require operands which are memory addresses. Some software simulators distinguish between instruction and operand addresses when setting and checking for trace and breakpoint control bits. This one does not. If an instruction accesses an operand in memory, the user may stop the simulation by setting a breakpoint at the instruction address or at the operand address. This allows the user the greatest amount of flexability.

Remember, the Trace and Execute (T,TA,TB,E,EA,TR) instructions deal with addresses and do not distinguish whether they are instruction or operand addresses.

Input/Output Simulation

The Simulator provides complete control over any simulated I/O that occurs during program execution. In addition to controlling the source and destination of data for the standard input and output ports and the serial port, the user may define any number of memory locations to be I/O locations and control the source and destination of data for these ports. Memory locations are declared to be I/O locations through use of the MIB, MIC, MIP, MIS, MOC, MOP, or MOS commands.

Input data values for standard input or memory mapped input ports may be obtained from the standard input device, from the current input device (controlled by Read commands), from a predefined data value, or from an input data buffer. The user may specify the source of input data for each input port.

Output data values from standard ports or memory mapped output ports may be written to the standard output device, the current output device (controlled by Write commands), or to an output data latch which can be examined with the DOUT, DIM, or DM commands. The user may specify where the output data is to be written for each output port.

Port Input

Requests for input data from the terminal by an input instruction will be indicated by the following message:

```
     *PPPP  INPUT ON PORT N =
where:  PPPP - is the porgram counter and
           N - is the port number
```

This message is only displayed when input data is requested from the standard input device and only in the interactive mode. If the input is requested at a memory mapped port, the message would read " PPPP INPUT ON MEMORY PORT N = ". Likewise, for the serial port, the message would be " PPPP INPUT OR SERIAL PORT = ". Any valid expression may be entered as the input value. An invalid expression or a value greater than 255 will cause the input message to be displayed again. The invalid input data is ignored. A blank line or no input (carriage return) response to the message will cause the Simulator to stop program execution and return control to the command mode. This feature may be used to advantage in certain situations.

Data values read in the batch mode from the standard or current input device or from the current device in the interactive mode must be supplied in the command stream where required. For example, assume the program is reading commands from the current input device in the batch mode and a "T 50" command is encountered, causing 50 instructions to be traced. If five input instructions are executed in these 50 instructions that request data from the current input device, these input values must follow the "T 50" command. When data values are supplied in this manner, more than one value may be specified on a line provided the data values are separated by commas. The following two methods of providing input data values are equivalent:

```
T 50                  T 50
20,40,50,100,10       20
                      40
                      50
                      100
                      10
```

If the user specifies an input port as begin preset (IP, SIP, or MIP commands), all data read from this port will be obtained from the predefined input latch and no message will be displayed asking for input data. The input latch value may be set by the SIN, SET SID, or SM commands. Note, the input port latch for memory mapped input is the memory location. This input mode is particularly useful for input data whose values do not typically change during simulation, such as the status of a UART.

Input data values may also be read from an input data buffer (IB, SIB, and MIB commands). Each request for input data reads the next value in the buffer associated with that port. When all of the data values have been used, the values are used again. A user may thus supply a recurring sequence of data values for a particular port. See the DATA command for a further discussion of this capability.

Regardless of the type of input port specified, the last input value for a port is saved in the input port latch. This value may be examined by the DIN command.

Port Output

When an output instruction is executed, the following message is displayed:

```
     *PPPP OUTPUT ON PORT N = VV
where:  PPPP - is the program counter and
           N - is the port number and
          VV - is the port value
```

As with thc corresponding INPUT message, slight variations indicate if the input is from a memory mapped port or a serial port. The user controls whether this message is written to the current output device (OC, SOC, and MOC commands) or the standard output device (OS, SOS, and MOS commands).

The user may specify an output port as latched only (OP, SOP, and MOP commands). In this case all data output written to this port will be placed into the output port latch and no message will be displayed. Note the output port latch for memory mapped output is the memory location.

Regardless of the type of output port specified by the user, the last output value is always saved in the output port latch.

This value can be examined by the DOUT, DIM, or DM commands and may be modified by the SOUT, SET SOD, or SM commands.

Input Errors

The response to input data errors is dependent upon the Simulation mode. In the interactive mode, if input was requested by the input message and an input error occurs, the message will be displayed again. If input is being read from a device other than the standard input device in the interactive mode and an error occurs, an error message will be displayed and the Simulator will return to the command mode. If an error occurs in the batch mode, an error message will be displayed and the program will terminate.

File Input

The RD and RED (read with delay) commands have been implemented so that the user may specify that I/O input data is to be read from an alternate I/O device, and then start program execution before the device switch is made. This would be done as follows:

```
RD      5
T       100
```

In this case the user has specified that additional input should be read from I/O device 5. It is assumed that this file probably contains input data. The user then specifies that 100 instructions should be traced. If the read command had gone into effect immediately, the user would not have been able to start instruction execution except by having the T command as the first command in the input stream on device 5. If the user had merely wanted to read commands from an alternate I/O device, the following command could be specified:

```
R       1
```

Interrupt Simulation

The Simulator allows the user complete freedom when simulating Normal 8080, Restart, or Trap Interrupts. An interrupt can be initiated after a certain number of cycles, or an interrupt can be initiated at a particular address.

As with the actual 8080/8085 microprocessor, the response of the Simulator to an interrupt is dependent on the internal enable bit, IE, and the interrupt mask bits. These bits are set and reset by microprocessor instructions just as they are in the actual microprocessor. These bits may also be initialized by the SET Command.

Interrupt Simulation is explained in detail in the description of the INT and NINT commands.

Standard Display Line

Throughout this manual, reference is made to the "Standard Display Line." This is the line that is displayed when the user is tracing through a program or uses the DC command. An example of the standard display line is shown below, preceded by a heading which is controlled by the H command:

```
 PC       INST            EA  (EA) NPC    CZSPI   A  B  C  D  E  H  L     SP    CYC
0001   STAX D            5302 25   0002   0000C   25 56 05 58 02 56 07   4041   0576
```

The standard display line consists of the following information. This information is displayed after the instruction whose mnemonic is displayed is executed.

PC - address of instruction just executed
INST - instruction mnemonic
NPC - address of the next instruction to be executed
EA - Instruction Operand, or Effective, Address
(EA) - contents of Effective Address
C - Carry flag
Z - Zero flag
S - Sign flag
P - Parity flag
I - Interdigit carry (half carry)
A - A Register
B - B Register
C - C Register
D - D Register
E - E Register
H - H Register
L - L Register
SP - stack pointer
CYC - cumulative cycle count

The following line shows the short format of the standard display line. This form of the standard display line is listed when the "FORM S" command is specified. The line consists of the Program Counter, the Instruction Mnemonic, and Register A.

```
0000  MVI  A,01     01
```

Character Set

The following list describes the characters that the simulator will recognize. Use of any other characters will cause the simulator to generate errors. Most of the special characters have no particular meaning in the simulator and may only appear within quote marks to denote an ASCII character.

Alphabetic Characters

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Numeric Characters

0 1 2 3 4 5 6 7 8 9

Special Characters

b̸	Blank Character
>	Greater Than
<	Less Than
'	Single Quote
,	Comma
+	Plus
-	Minus
/	Slash
$	Dollar Sign
*	Asterisk
(	Left Parenthesis
!	Exclamation
)	Right Parenthesis
.	Period
&	Ampersand
"	Double Quote
#	Sharp
%	Percent
:	Colon
;	Semi-colon
=	Equal
?	Question Mark
@	At Sign
	Tab

Constants

A constant is an invariant quantity which may be an arithmetic value or an ASCII character code. There are several ways of specifying constants in the simulator.

Decimal constants can be defined as a sequence of numeric characters optionally preceded by a plus or a minus sign. If unsigned, the value is assumed to be positive. Other constants are defined by placing a one letter descriptor after the constant. If the descriptor is hexadecimal, a leading Ø must be added to values that start with A-F (unless the user has specified the "BASE HS" command). This will distinguish a hexadecimal number from a symbol. The legal descriptors and their corresponding bases are shown below. If no descriptor is given, the number is assumed to be decimal.

B - Binary
O - Octal
Q - Octal
D - Decimal
H - Hexadecimal

An ASCII character constant may be specified by enclosing a character in single quotes. (For example, 'A'.) The character constant may be used anywhere that a numeric constant may be used.

Through use of the BASE command, the user may specify that all numeric constants are in hexadecimal. This is useful for those who debug their programs in hexadecimal, since it makes it unnecessary to specify the "H" after each constant. (See BASE command for further details.)

Symbols

A symbol is a sequence of characters, the first of which must be alphabetic or one of the special characters ? or @. Except for these two special characters, only alphanumeric characters may be used in a symbol.

Only the first six characters of a symbol define the symbol and are retained by the Simulator in the symbol table. Additional characters may be added to a symbol for documentation. The parameter in the program that dictates the length of a symbol may be changed by the user at compile time (see Installation Notes).

Typically, a user will use those symbols that were defined during the assembly of the program being simulated and that were read into the Simulator along with the object data. However, a user may define new symbols or change the value of a symbol with the SSYM command.

Since some assemblers and higher level languages allow the definition of the same symbol more than once in a program (in macros, for example), a method is required to uniquely specify such a redundant symbol to the Simulation program.

The Simulator enables the user to uniquely specify a redundant symbol by allowing the specification of a "symbol string" in place of a symbol. For example, assume that the label "CALR" is a valid, though redundant, symbol, and the user wishes to refer to a particular occurence of that symbol in the program. This may be done by specifying a unique symbol string in a command argument consisting of one or more symbols which preceed the symbol of interest in the symbol table.

In the example just mentioned, assume that the symbol "LOOP2" preceeds the symbol "CALR" in the symbol table at the occurrence the user wishes to refer to. Then specifying "LOOP2/CALR" will uniquely specify the occurence of the symbol "CALR" desired. Slash characters are used to separate, concatonate, symbols in a symbol string. Such a concatenated symbol string may be used anywhere a simple symbol is permitted. A symbol string of the form "CALR/CALR" is permitted and would indicate the second occurrence of the symbol "CALR". The DSYM command may be used to display the symbol table and determine the sequence of symbols in the table. Typically, duplicate symbols will not be present, and a single symbol will be sufficient to uniquely define that symbol.

The general format of the symbol string is:

$\mathrm{sym}_1/\mathrm{sym}_2/\mathrm{sym}_3/ \ldots /\mathrm{sym}_n \pm \mathrm{expression}$

This causes a search for sym_1 followed by a search for sym_2 and so on until sym_n is found. A symbol string can be used in any expression that a single symbol may be used since it evaluates to a single symbol value. A constant may be added or subtracted from the final symbol in the concatedated string as if the string were a solitary symbol.

Program Counter

By use of the symbol "$", the user may include the current value of the simulated program counter in any expression. "$" always represents the address of the next instruction to be executed or the new value of the program counter if it has been modified by the SET command. For example, the following commands are valid:

```
SET  PC=$+20
RA   $+2 10
```

Expressions

An expression is a sequence of one or more symbols, constants, or the location counter symbol, "$", joined by the arithmetic operators + and -. Parenthesis are not allowed and all expressions are evaluated from left to right.

Expressions may be used anywhere a numeric value is required. All arithmetic is performed using 16 bit values and hence all values are modulo 65536.

Range Lists

Many of the simulator commands accept operands that may consist of a single value or a contiguous range of values. This is called a Range List. Typically, a Range List specifies a range of addresses for a command. For example, to display a group of memory addresses, the user may type:

```
DM    Ø  ØFFH
```

The address range specified is a Range List. In general, a Range List consists of a single expression or two expressions without a separating comma. Thus the above command will display all memory locations from Ø to ØFFH , while the command

```
DM    Ø, ØFFH
```

would display only locations Ø and ØFFH. From the above examples it can be seen that a separating comma determines whether a range list consists of a single value or a range of values. It should be noted that the comma must immediately follow the first value but blanks may exist between the comma and successive values.

In a range list, the second expression, if present, must be a value greater than or equal to the first expression, otherwise an error message will be generated.

Remember, commas separate ranges and blanks separate values within a range.

SIMMULATOR COMMANDS

This section describes the simulator commands. A command begins in the first column of the input line. Only one command may be placed on a line; however, many commands allow multiple arguments. At least one blank or tab must separate the command from any operands. In general, command operands may be separated by blanks, tabs, or commas. For some commands a blank or tab as the separator will perform a different function than a comma. Remember that range lists are separated by a blank or a tab. Individual addresses are separated by commas. Multiple command arguments are separated by commas.

The following nomenclature is used in the command descriptions:

{ } - denotes an optional operand or part of the command name

R - indicates a Range List

A - denotes a memory address

N - denotes an expression

V - denotes an expression and typically represents a byte value

* - Comment Line
;

A comment may be included among Simulator Commands by placing an "*" or ";" as the first character on a command line. In some cases a comment may be included on the same line as a command but only if the command requires a definite number of parameters and they have all been specified.

Comments are useful to document and describe blocks of commands or data values that seldom change.

Example:

```
     *  THIS SIMULATION TESTS THE BINARY TO BCD CONVERSION
     ;  PROGRAM FOR THE Z80 MICROPROCESSOR
```

BASE - Set Numeric Input Base

BASE {D,H,HS}

All numeric values specified as input data or command arguments are assumed to be decimal unless a descriptor is used to indicate a different base. The user may specify hexadecimal, decimal, octal, or binary be placing the descriptors H, D, O or Q, or B after the value. For example: 37Q.

The BASE command may be used to specify that all numeric values will be treated as hexadecimal values. There are two hexadecimal modes that can be specified. The "H" operand specifies that all input values will be treated as hexadecimal; values that start with A-F must begin with a zero in this mode. The "HS" operand specifies that all input values will be hexadecimal and, in addition, the values do not have to start with a leading zero. In this case, input data values beginning with A-F are first assumed to be symbols. If no corresponding symbol is in the symbol table, the input data is assumed to be a numeric. If the base is set to either the "H" or "HS" mode, the descriptor H after a numeric data value is optional. Thus 1FH could also be specified as 1F.

The "D" command argument may be used to switch back to the decimal default mode.

Note, when in one of the hexadecimal modes, values other than hexadecimal may not be entered by appending a descriptor after the value. Except for the descriptor "H", any other descriptor will either cause an error to be generated or cause the input data to be recognized as a value not intended by the user.

Example:

```
BASE   D
BASE   HS
```

Error Conditions:

1. Operand specified and not D, H, or HS

BP — Set Breakpoint
NBP — Clear Breakpoint

```
{N}BP    {R {,R,  R, ...}}
```

The BP and NBP are used to set or clear an instruction or operand breakpoint. During execution of instructions by the E, EA, or TB commands, the encountering of a breakpoint will cause program execution to terminate. The standard display line is then displayed. Breakpoints may be set for any memory location. The memory location may contain an instruction, an instruction operand, or may even be an I/O location.

The BP command is used to set a breakpoint at an address or a range of addresses. The NBP command enables the user to negate the effect of the BP command. Any addresses specified in the NBP command will have their breakpoint flags permanently cleared so that no breakpoint will occur when these addresses are accessed.

These commands may also be specified without any operands. In this case, the command affects only the master breakpoint flag. When the NBP command is used without any operands, all checks for breakpoints are inhibited during the E, EA, or TB commands, but the breakpoint flags previously set will remain set. The BP command with no operands may then be used to reactivate the breakpoints. This feature is useful when the user thinks a section of code is completely debugged and ready for final checkout, but is not completely sure the code is valid for all possible inputs. The user may turn off breakpoints, run several test cases, and, if a problem is encountered, turn the master breakpoint flag back on for further debugging.

Example:

```
NBP
BP     77H, 1ØØH 1ØFH,5
NBP    1BH
```

Error Conditions:

1. Invalid operand
2. Ending address less than starting address in range list

DATA — Specify Input Buffer Data
MDAT
SDAT

```
DATA     {*}
DATA     N  {V, V, ...}
SDAT        {V, V, ...}
```

The DATA, MDAT, and SDAT commands are used to enter data into the Simulator's input data buffer. This FIFO buffer may be used to supply data values as required to any of the input ports. The DATA directive specifies values for normal ports, the MDAT directive specifies values for memory mapped I/O ports, and the SDAT directive specifies values for the serial port. A port can be made to read data from the Data Buffer through the use of the IB, MIB, and SIB commands. The DATA, MDAT, and the SDAT commands may also be used to vary the parameters associated with the input data buffer. The commands may be used in the ways described below to perform the stated functions. The port number parameter, N, is not specified for the SDAT command.

1. If no argument is specified, the input data buffer table is cleared of data for all ports. This variation of the command is typically used when the user wishes to change the data in the buffer. The user would specify the command without any operand followed by the command with operands. The second command would be used to fill the buffer with data. Any of the three commands may be used to clear the buffer for all types of ports.

2. If an "*" is specified as the operand, then any port requesting data will obtain the first data value associated with the port. In this case, the "pointers" associated with each port are reset to point to the first data value entered.

3. "DATA N" acts in a similar manner to "DATA *" except only the "pointer" for port N is reset. This form does not apply for the SDAT command.

4. "DATA N V ..." is used to enter data values for port N into the buffer. The order in which the data is entered is the same order in which the data will be "read" by microprocessor input instructions. Therefore, the buffer is refered to as a FIFO (first-in-first-out) buffer.

Input data values for different ports may be entered into the buffer in any order; all data for a particular port does not have to be entered consecutively. The user may enter 5 bytes for port Ø followed by 3 bytes for port 2 followed by another 2 bytes for port Ø. Only the data associated with a particular port can be "read" by that port.

If the program "reads" more data values from a port than have been entered into the buffer, the data values specified will be used again, starting with the first data value entered for that port. In other words, the data "pointers" are automatically reset when necessary. This feature can be advantageous when an input port supplies the same data values repeatedly.

If the program being simulated attempts to read input data from the buffer and no data has been entered for that port, a warning will be printed. The contents of the registers will not change.

No data values will be entered into the buffer unless the input line is error free. This avoids the problem of the user knowing if any of the input data values were entered when an input error occurs.

Example:

```
DATA
DATA    *
DATA    1  45,6FH,Ø, Ø
DATA    7
MDAT    1ØØØH  25,10
SDAT    1  0  0  1  1
```

Error Conditions:

1. Port number greater than 255, 65535 for memory ports
2. Data value greater than 255, 1 for serial port
3. Invalid operand
4. Data buffer filled

DC — Display CPU

DC {*}

This command is used to display the standard display line immediately. The long form of the display line is used even if the command "FORM S" has been specified. This command is typically used after instruction execution with the short display line or if the user is using a terminal and the listing is routed to another device, making the results of the last instruction execution unavailable.

The line displayed by the DC command will always contain the address and instruction mnemonic of the last instruction executed. If the user has modified the program counter, execution will commence at the new address displayed in the NPC field. Any modifications to the other elements of the display line (PSW, registers, etc.) will be immediately reflected by the command.

The registers displayed by this command will be from the currently selected register bank. The user may examine the alternate register bank values by specifying "*" as the command operand.

The DC command does not modify any of the heading parameters or counts. A heading is never displayed with this command.

Example:

```
     DC
     DC      *
```

Error Conditions:

1. Invalid operand

DEL - Delete Symbols

```
DEL   {symbol string{,symbol string, ...}}
```

The DEL command is used to delete a few symbols from the symbol table or to delete all symbols from the symbol table. If no operand is specified, then all symbols in the table are deleted. If a symbol(s) is specified, only that symbol(s) is deleted from the symbol table.

A deleted symbol will provide additional room in the symbol table, which may be of advantage if the user has encountered the "SYMBOL TABLE FULL" message.

Example:

```
     DEL
     DEL    TABLE, LABEL/ONE
```

Error Conditions:

1. Invalid symbol string

DH - Display History

```
DH    {V}
```

The DH command may be used to display the addresses of previous instructions executed by the simulator. "V" instruction addresses will be displayed. Each line of the display consists of the address of the instruction executed, starting with the instruction executed "V" instructions ago.

If no operand is specified, 32 instruction addresses will be displayed. If the operand is greater than 32, only 32 instruction addresses will be displayed. An example of the DH command may be found in the sample simulations.

At the start of simulation, if "V" is greater than the number of instructions executed, "****" will be displayed for instruction addresses not executed.

Example:

```
     DH   7
```

DIN — Display Input Port
DOUT — Display Output Port

```
DIN       R  {,R, R, ...}
DOUT      R  {,R, R, ...}
```

The DIN and DOUT commands are used to display the contents of the processor input and output ports. The last value read from an input port is always saved in the input port latch and may be examined by this command. The DIN command also allows the user to examine the value to which an input port has been preset by the SIN or FIN command.

The last value written to an output port is saved in the output port latch and may be examined by the DOUT command. This command also allows the user to display the value to which an output port has been set by the SOUT command.

The maximum value of any operand is 255.

Example:

```
     DIN   Ø,1
     DOUT  Ø 16
```

Error Conditions:

1. Operand not specified
2. Port number out of range
3. Invalid operand

DM — Display Memory
D

```
D{M}    R  {,R, R, ...}
```

This command is used to display the contents of the simulated memory. The operands are ranges of addresses which are to be displayed. Each range will be displayed starting on a new line. Up to 16 bytes will be displayed on each line. An example of the DM command may be seen in the sample programs.

The maximum value of any operand may be ØFFFFH or the maximum memory size set at compile time if smaller.

This command may also be used to examine memory mapped input port preset values or the last value read at a memory mapped port that is not preset. Likewise, the command can be used to examine the last value to be written to a memory mapped output port.

Example:

```
     DM  Ø  3FH,1ØØH, 2ØØH
     DM  3ØØ  3FF
```

Error Conditions:

1. Operand not specified
2. Address out of range
3. Invalid operand
4. Ending address of range less than starting address

DIM — Display Interrupt Mask

DIM

This command is used to display the Interrupt Status and serial input bits, IM, read by the RIM instruction; the serial output bit, SOD; and the edge triggered Restart 7.5 flipflop.

The output generated by this command is shown below.

```
IM = 10000000    SOD = 1  INT7.5 = 0
```

The Binary bits shown in the interrupt mask correspond exactly to those read into the A register by the RIM instruction. The bits are:

Bit 7 - Serial input data
Bit 6 - Interrupt Pending, Restart 7.5
Bit 5 - Interrupt Pending, Restart 6.5
Bit 4 - Interrupt Pending, Restart 5.5
Bit 3 - Interrupt Enable Flag
Bit 2 - Interrupt Mask Bit, Restart 7.5
Bit 1 - Interrupt Mask Bit, Restart 6.5
Bit 0 - Interrupt Mask Bit, Restart 5.5

DSYM - Display Symbols

```
DSYM     {symbol string  {,symbol string, ...}}
```

This command is used to display the value of a symbol or the values of all the symbols in the symbol table. If no operand is specified, then each symbol in the symbol table along with its value is displayed, one symbol per line. If operands are specified, each symbol specified along with its value is displayed.

If there are no symbols in the symbol table, the command with no operand will display no information. If a symbol is specified but it is not in the symbol table, the message "UNDEFINED SYMBOL" will be displayed.

Example:

```
     DSYM      START,TABLE
     DSYM      GO/DATA
     DSYM
```

Error Conditions:

1. Undefined symbol present in argument list
2. Invalid symbol terminator

E - Execute Instructions
EA - Execute Instructions until Address

```
E       {N}
EA       A  {N}
```

The E and EA commands cause the program to begin execution of instructions. The execution begins at the address contained in the simulated Program Counter. The standard display line is not displayed for instructions executed unless the trace flag for that instruction has been set by the TR command. This distinguishes these commands from the T and TA commands.

These commands will terminate program execution and display the standard display line for the final instruction executed if any one of the following conditions is met:

1. A breakpoint is encountered
2. An illegal instruction is executed
3. The number of instructions specified by the LI command is executed.

For the E command, the optional "N" parameter specifies the number of instructions that should be executed before the command terminates. This value overrides the value specified by the LI command.

The EA command is similar to the E command except that the program will continue to execute instructions until the instruction at address "A" is executed. If "N" is also specified, the instruction at address "A" must be executed "N" times before the command terminates. In either case, the EA command will also terminate program execution if any one of the three conditions mentioned above is met.

Example:

```
E    20
EA   3FH  4
```

Error Conditions:

1. Address not specified for EA command
2. Invalid operand specified

FIN — Fill Input Port
FOUT — Fill Output Port

FIN A_s A_e V {,A_s A_e V, ...}

FOUT A_s A_e V {,A_s A_e V ...}

The FIN and FOUT commands are used to fill a group of input or output ports will a specified value. "A_s" specifies a starting port number and "A_e" specifies the ending port number that will be filled with the value "V". All ports starting at "A_s" up to and including "A_e" will be set to the value "V". As many port ranges as desired may be set to the specified values with a single command.

The maximum value that may be specified for a port number is 255. The maximum value of the data that can be placed in a port buffer is ØFFH.

The FIN command may be used to specify the Preset value used when an input port is specified as preset with the IP command. Although no microprocessor instruction can read the values set into an output port latch, the user may wish to initialize these values with the FOUT command before executing a section of code that writes data to these ports.

Example:

```
FIN   Ø 7  ØFFH
FOUT  4 7  3, 0 2  4FH
```

Error Conditions:

1. Port number too large
2. Ending port number less than starting port number
3. Port value too large
4. Operand error

FORM - Set Display Line Format

```
FORM   {L}{S}
```

The standard display line consists of the Program Counter, instruction mnemonic, Next Program Counter, status bits, registers, and the cumulative cycle count. For users in an interactive mode and with slow terminals, the listing of a standard display line requires a reasonable amount of time and contains more information than is needed. These users may use the FORM command to turn on the short display line listing option. The short standard display line consists of the Program Counter, instruction mnemonic, and the A register. Whenever a standard display line is required, only this information is displayed. The one exception to this is the DC command, which always displays the long form of the standard display line.

When it is necessary only to follow the flow of the program and the values of the registers are not of interest, the short display line format is particularly useful.

Users may modify the information that is displayed with the short display line option to suit their particular needs. How to modify the contents of this line is discussed in the Simulator Installation Notes.

The "L" operand requests the long display format while the "S" operand requests the short format. The default is the long format.

Example:

```
FORM S
FORM L
```

Error Conditions:

1. Operand not specified
2. Operand not L or S

FM - Fill Memory

FM A_s A_e V {,A_s A_e V, ...}

The FM command is used to fill a range of memory locations with a specified value. "A_s" specifies the starting memory address and "A_e" specifies the ending memory address that will be filled with the value "V". All memory locations starting at "A_s" up to and including "A_e" will be set to the value "V". As many blocks of memory as desired may be set to a given value with a single command. This command is useful when the user desires to read a new object module into simulated memory after already having done some simulation with a different object module. In this case, the user could fill the complete Memory with the halt opcode value. The simulator initially sets the Memory to this value so that, if the program counter gets out of range, the program will halt.

The maximum value that may be specified for any memory addresses is ØFFFFH or the maximum memory size set at compile time, if that is smaller (see Installation Notes).

Example:

```
    FP  Ø OFFFFH   76H
```

Error Conditions:

1. Memory Address too large
2. Ending address less than starting address
3. Memory value too large
4. Operand error

H - Specify heading count

H {N}

This command is used to specify the heading display count. The heading, which can be seen in the sample program, describes the information on the standard display line.

If no operand is specified on the command, then no headings will be displayed during further instruction execution and listing. If the heading count is specified as Ø, then a heading is generated immediately but no other parameters set by previous H commands are affected. "H 0" is typically used when the user has turned the heading off, but when the user would like a single heading before generating trace information.

If the heading count is greater than Ø, a heading will be displayed after every Nth instruction has been traced. The default is "H 1Ø".

Example:

```
H
H   25
```

Errors:

1. Invalid operand specified

IB — Read Port Input Data from Data Buffer

IC — Read Port Input Data from Current input device

IP — Read Port Input Data from preset data latch

IS — Read Port Input Data from Standard input device

```
Ix     R  {,R, R, ...}
```

These four commands allow the user wide flexibility in the simulation of microprocessor input instructions. Each port may "read" its data from one of the sources described below. Of course, the source of a port's input data can be altered at any time during the simulation.

IB - any port specified by this command will "read" data from the input buffer table (see DATA command). If more data is "read" than has been entered in the table, the data is re-read. An attempt to read from the buffer, by a port for which no data has been entered, will result in a warning message. In the batch mode, the input instructions will be executed but the accumulator will not change. If in the interactive mode, the Simulator will return to the command mode.

IC - a port specified in this command will "read" data from the current input device. This device may be either the standard input device or a device or file specified by the R or RE commands. If the current device is the same as the standard device, this command has the same effect as the IS command.

IS - a port specified in this command will "read" data from the standard input device that was set at compile time. If the program is in the interactive mode, the following message will be displayed to request the input value from the user:

```
*PPPP   INPUT ON PORT N =
```

where PPPP is the address of the input instruction and N is the port number. If the program is in the batch mode, no message will be displayed. In the batch mode, data bytes must be included in the command stream where required.

IP - a port specified in this command will "read" data from the preset data latch. The value in this latch may be specified by the SIN command. This input mode is typically used for those ports which contain data which will not change during the simulation, such as an I/O status value.

When the Simulator is first entered, all ports are initialized as though they were set by the IC command. When reading data in the batch mode or in a read mode, from a port set by the IC command (also by IS in batch mode), the user must provide the data values in the command stream where needed. For example, if a T or E instruction causes five input values to be read, these five values must follow the T or E command. The user may specify more than one data byte per line. The data bytes are separated by blanks.

If an End-of-file (EOF) condition is encountered while reading input data for ports in the IC or IS mode, the Simulator will return to the command mode. This is especially useful in the interactive mode as a way to stop program simulation. If invalid data is read by the Simulator or a value is out of range, the response will depend upon the mode of the Simulator. If the user was prompted by the message requesting input data as shown under the IS command description, the user will be prompted again for the correct data. If the user was in any other mode, an error message will be displayed and the Simulator will return to the command mode. An error does not cause the program counter to be updated. This allows the user to easily continue processing at the same input instruction.

Example:

```
IB   2
IS   0 2,7
```

Error Conditions:

1. Input port greater than 255
2. No operand specified

INT — Set Interrupt
NINT — Clear Interrupt

```
INT     type      cycles     {instruction or address}
NINT
```

These commands may be used to simulate the interrupt mechanism of the microprocessor. Normal 8080, Restart, or Trap interrupts can be simulated. Only one interrupt may be pending at any time. If the INT command is used to specify an interrupt, any previously specified interrupt that has not occurred is cancelled. For a Normal 8080 or Restart interrupt to be recognized, the interrupt enable bit must be set. For a Restart interrupt to be recognized, the interrupt must not be masked. If these bits are not set, the interrupt will not be recognized at the specified time. However, the interrupt will still be pending and, unless cleared, will occur as soon as the interrupt enable bit is set.

"Type" specifies which kind of interrupt is to occur:

I	-	Normal 8080 Interrupt
T	-	Trap Interrupt
5	-	Restart 5.5 Interrupt
6	-	Restart 6.5 Interrupt
7	-	Restart 7.5 Interrupt

"Cycles" specifies the number of cycles after the current instruction at which the interrupt is to occur. A cycle count of Ø will cause an interrupt to occur immediately, as if the interrupt had actually occured during the previous instruction execution. Remember, if an interrupt occurs during the execution of an instruction, that instruction execution is completed before the interrupt is recognized.

For a Normal 8080 interrupt, the user may specify any 8080/8085 instruction with the interrupt command. The bytes of multi-byte instructions are separated by blanks. The instruction field is ignored for Restart and Trap interrupts.

If no instruction or data byte is specified, the last one specified is used.

As with the actual 8080/8085, the interrupt enable bit is reset when the Simulator is reset and when the simulation program is first executed.

Although the interrupt simulation mechanism is based on a cycle count, it is also easy to simulate an interrupt at a particular address. The user may set a breakpoint at the address he wishes to simulate the interrupt and when that address is reached, he can specify an interrupt command with a cycle count of 0.

Examples (Mode 0 assumed):

INT	I	25	C7	Perform Normal 8080 interrupt after 25 cycles, then execute a RST 8 instruction
INT	6	0		Perform Restart 6.5 interrupt immediately
NINT				Clear pending interrupt

Error Conditions:

1. Invalid Operand
2. Operand Not Specified

L — Load Object Module
LS — Load Object Module with Symbol Table

L{S}   {*} {I/O device} {file name}

The L command is used to load an object module into simulated memory while the LS command is used to load an object module into simulated memory, and additionally, load any symbol table information present in the object module. Note that although symbols may be present in an object module read by the L command, these symbols will not be placed into the symbol table. This feature is useful since the symbols in an object module are placed into the symbol table even if the same symbol already exists. Thus the L command avoids having many duplicate symbols, reducing the possibility of symbol table overflow.

The object module may be read into simulated memory from the logical I/O device number or file name specified in the operand field. The operands are:

* - the object module is read from the standard object module unit that was specified in the program at compile time. This would typically be a paper tape reader or some default file name. This method avoids the problem of users having to know the device numbers of peripherals.

I/O device - this is a numeric value which specifies that the object module will be read from the logical unit specified.

file name - specifies that the object module will be read from the file specified. If the file does not exist, the message "FILE NOT FOUND" will be printed.

If the user does not specify any operand, it is the same as if "*" was specified. After reading the object module, the program will display the message "NUMBER OF BYTES READ =   ". If the number of bytes read is Ø, it is possible that the information read was not in the proper format for an object module. In this case the message

"END OF FILE ENCOUNTERED" is displayed. The Simulator's Program Counter will be set to the starting address specified in the load module.

Note that in the batch mode, both commands and object module may be read from the same unit.

Example:

```
L       *
LS      5
L       TESTZ8
```

Error Conditions:

1. Object module contains invalid hexadecimal characters
2. Object module contains invalid symbol information
3. Symbol table overflow. In this case, all remaining symbols in the object module are ignored and all data is processed and placed into simulated memory.
4. Checksum error encountered in object module
5. Address out of range. A load address specified in the object module was larger than the simulated memory.

LI - Set Instruction Execution Limit

```
LI    N
```

This command is used to specify a limit to the number of instructions that will be executed during an E, EA, T, TA, or TB command. When this limit is reached, the message "LIMIT REACHED" is displayed and control returns to the command mode. This limit may be changed for the duration of the command by specifying an optional limit on the E, T or TB commands. The above message is only displayed when no limit was specified on the E, T, or TB command and the limit set by this command has been reached. There is no inherent limit parameter for the EA or TA commands, so the only limit which applies is set by this command.

This limit applies only for the duration of the current command and is not cumulative for all commands. When running in the batch mode or a read mode, it may be necessary to increase the limit to enable a large program to execute to completion with one E or T command. The default for this command is 1000 instructions.

Example:

```
     LI     100
```

Error Conditions:

1. No limit specified
2. Error in limit specified

MIB - Read Memory Mapped input data from Data Buffer

MIC - Read Memory Mapped input data from Current input device

MIP - Read Memory Mapped input data from preset data latch

MIS - Read Memory Mapped input data from Standard input device

```
MIx    R {,R, R, ...}
```

These four commands allow the user wide flexability in the simulation of Memory Mapped Input. Each Memory Mapped Input Port may "read" its input data from any one of the sources described below. Of course, the source of a port's input data can be altered at any time during the simulation.

The memory mapped input commands, MIB, MIC, MIP, and MIS are analogous to the Normal Port Input commands, IB, IC, IP, and IS. A brief description of the Memory Mapped Input Commands is given here. The user may refer to the Port Input Command descriptions for more details.

MIB - read memory mapped input data from Data Buffer. Data is entered into the Data Buffer by the MDAT Command.

MIC - read memory mapped input data from the current input device.

MIS - read memory mapped input data from the standard input device. The following message is displayed at the standard input device in the interactive mode:

```
*PPPP   INPUT ON MEMORY PORT N =
```

where PPPP is the address of the input instruction and N is the port number.

MIP - read memory mapped input data from the value preset into the memory location. The value may be preset by the SM command. Note that memory not mentioned by any of the memory mapped I/O commands acts as though it were preset.

Simulated memory used as a Memory Mapped input port would act as a preset port if no memory I/O instruction were specified. Therefore, specifying the MIP command for a memory location not previously specified as a Memory Mapped input port would have no effect. The MIP command may also be used to turn a Memory Mapped I/O port back into regular memory location.

There is no limit to the number of memory locations that can be declared to be I/O ports.

The actions taken when an input data error is encountered are the same as those actions taken for normal input port errors. These error actions are discussed in the Normal Port I/O Command descriptions (IB, IC, IP, and IS).

Examples:

```
MIC     5000H,5010H
MIB     5050H
```

Error Conditions:

1. Port number greater than 65535
2. No operand specified

MOC — Write Memory Mapped Output Data to Current Output Device
MOP — Write Memory Mapped Output Data to Memory Location
MOS — Write Memory Mapped Output Data to Standard Output Device

```
MOx     R   {,R,R, ...}
```

These three commands allow the user wide flexability in the simulation of Memory Mapped output. Each Memory Mapped output port may "write" its data to one of the destinations described below. The destination of a memory mapped port's output data may be changed at any time during the simulation.

The memory mapped output commands, MOC, MOP, and MOS, are analogous to the Normal Port Output Commands, OC, OP, and OS. A brief description of the Memory Mapped commands is given here. The user may refer to the Port Output Command descriptions for more details.

MOC - write output data to the current output device. The following message is displayed:

```
*PPPP    OUTPUT ON MEMORY PORT N = V
```

where PPPP is the address of the instruction writing the output data to the port, N is the port number, and V is the output data value.

MOS - write output data to the standard output device. The same message described in the MOC command description is displayed.

MOP - write output data to memory mapped port location only. Note that memory not mentioned by any of the memory mapped I/O commands acts as though it is set in this manner.

Simulated memory used as a memory mapped output port would act as a latched port (value written to memory location only) if no memory I/O instruction were specified. Therefore, specifying the

MOP command for a memory location not previously specified as a Memory Mapped Output port would have no effect. The MOP command may be used to turn a Memory Mapped I/O port back into a regular memory location.

There is no limit to the number of memory locations that can be declared I/O ports.

Examples:

```
    MOC      5001H,5011H
    MOS      5051H
```

Error Conditions:

1. Port number greater than 65535
2. No operand specified

OC — Write port output data to Current output device
OP — Write port output data to data latch
OS — Write port output data to Standard output device

```
Ox     R {,R, R, ...}
```

These commands allow the user wide flexibility in the simulation of microprocessor output instructions. Each port may "write" its data to one of the destinations described below. The destination of a port's output data may be modified at any time during the simulation.

OC - a port specified in this command will "write" data to the current output device with the following message:

```
*PPPP  OUTPUT ON PORT N = V
```

PPPP specifies the address of the output instruction writing data to the port, N is the port number, and V is the value written to the port.

OS - a port specified in this command will "write" data to the standard output device with the message shown for the OC command. This command is typically used when the user has specified the W command but would like to see the output data of any output instructions on the standard output device.

OP - a port specified by this command will "write" data to the output port latch only. The value in this latch may be examined by the DO command. This command is typically used when output occurs that is not of current interest to the user.

Note that the last value written to a port is saved in the output port latch regardless of the mode specified for the output port. At the start of the program all output ports are initialized as though they were set by the OC command.

Example:

```
OC  0,2
OP  5
```

Error Conditions:

1. Output port greater than 255
2. No operand specified

PRO — Protect Memory
NPRO

```
{N}PRO     R {,R,R, ...}
```

The PRO command allows the user to specify portions of memory that should not be written into (Simulated ROM). When an attempt is made to write into Protected Memory, an informative error message is displayed. The contents of the memory are not changed.

The NPRO command enables the user to negate the effect of the PRO command. The protect flag will be reset for the address range specified.

The PRO and NPRO commands may also be specified without any arguments. In this case, the commands affect only the master protect flag. The NPRO command without an argument turns off the master protect flag. Checking for protected memory will not be performed until enabled again by specifying the PRO command without any arguments. The PRO and NPRO commands do not affect the protect flags at specific addresses. They only turn the master flag off and on.

Examples:

```
PRO     100H 200H
NPRO    0    OFFFFH
```

Error Conditions:

1. Invalid Operand
2. Ending address less than starting address in range list

R — Read Commands
RD — Read Commands with delay
RE — Read Commands with echo
RED — Read Commands with echo and delay

```
R{D}     {*}  {I/O device}  {file name}
RE{D}    {*}  {I/O device}  {file name}
```

These commands enable the user to read subsequent commands or input data values from an alternate I/O device. The RE and RED commands will read the input data from the alternate device or disk file and also echo the input to the current output device. The RD and RED commands will not go into effect until one additional command has been entered on the current device. These commands may be used when reading input data from a file. The RD or RED command can be specified immediately followed by a trace or execute command. If there was not a one-instruction delay before the input device was switched, the first entry in the file of input data would have to be a Trace or Execute command. The R and RE commands are typically used to execute a complete set of commands that have been debugged and reside on a file.

Subsequent input may be read from the following sources when the argument underlined is specified:

* - read subsequent input from the standard input device specified in the program at compile time. This is typically a terminal in the interactive mode or a card reader in the batch mode. The command with this parameter is not usually used since all input is typically read from the standard input device, anyway. However, it may be used to echo commands to the terminal in the interactive mode or to not echo commands to the list device in the batch mode. In addition, in the interactive mode, using this command will cause the command prompt character not to be displayed.

I/O device - this is a numeric value that specifies a FORTRAN logical I/O unit from which subsequent input will be read.

file name - specifies that subsequent input will be read from the file specified. File names must begin with an alphabetic character. If the file does not exist, the message "FILE NOT FOUND" will be displayed.

If the user does not specify any operand, it is the same as if "*" were specified.

Example:

```
RE      *
RD      TESTFILE
R       5
```

Error Conditions:

1. File not found
2. Invalid I/O device number specified

RES — Reset Microprocessor

RES

The RES command is used to reset the Simulator in a similar fashion to activating the reset line on the actual device. The RES command performs the following functions:

1. Program counter is set to ØØØØ
2. Stack Pointer is set to Ø
3. Interrupt enable bit is reset
4. All restart interrupts are masked
5. All accessable Registers are set to Ø
6. Cycle count is set to Ø

After a RES command, if the user enters a DC command, the resulting output display will still show the address of the last instruction that was executed. However, the Next Program Counter (NPC) will contain a zero. The elements listed above will be set to the values specified above. The next instruction executed will be the one at location Ø.

RET — Return from Read Mode

RET

This command is used to restore the simulator input mode to the standard input device after an R, RD, RE, or RED command (Read Commands from file) has been specified. Thus the RET command should be the last command in a command stream read by one of the above commands. An End-of-File condition will have the same effect as the RET command. If this command is used when a read command is not in effect, no action takes place.

The RET command is similar to the "R *" command. The "R *" command returns control to the standard input device as does the RET command. However, as will all R commands, the "R *" command prevents the prompt character from being generated in the interactive mode. In contrast, the RET command exits the read mode and displays the prompt character in the command mode.

S — Set Processor Element
SET

```
S {ET}    ele=V   {,ele=V,ele=V,...}
```

The SET command is sued to set the values of the various registers and status bits of the microprocessor.

The elements, along with the legal maximum values, are listed below:

A	-	Register A	(255)
B	-	Register B	(255)
C	-	Register C	(255)
D	-	Register D	(255)
E	-	Register E	(255)
H	-	Register H	(255)
L	-	Register L	(255)
Z	-	Zero Flag	(1)
CY	-	Carry Flag	(1)
P	-	Parity/overflow Flag	(1)
S	-	Sign Flag	(1)
I	-	Interdigit Carry, Half Carry	(1)
SP	-	Stack Pointer	(65535)
CC	-	Cycle Count	(65535)
PC	-	Program Counter	(65535)
IE	-	Interrupt Enable	(1)
IM	-	Interrupt Mask	(7)
I7	-	Restart 7 Flip Flop	(1)
SI	-	Serial Input Latch	(1)
SO	-	Serial Output Latch	(1)

Most of these elements can be displayed through use of the DC and DIM commands.

The equal sign between the elements and their values is optional. If desired, it can be replaced with a blank.

Examples:

```
SET    A=45H,C=55,PC=200H
SET    IE=1
```

Error Conditions:

1. Invalid elements specified
2. Invalid separator after element
3. Element value out of range

SIN — Set Input Port
SOUT — Set Output Port

```
SIN   A   V {,V, V, ...}
SOUT  A   V {,V, V, ...}
```

The SIN and SOUT commands are used to set and/or examine the value of the processor input and output ports respectively. The first operand of these commands specifies a port number at which the following data will be entered or examined. The first data byte (V) will be entered at the specified port number and successive data bytes will be entered at successive ports.

The user may continue this command on additional lines by terminating the last data value on a line with a comma. If the command is continued, the address of the next I/O port, followed by the contents of that port, will be displayed on the following line. For example:

```
ØØØ1  Ø5 -
```

The user may then modify the contents of this port as well as the contents of successive ports as required, starting at the port number displayed. If a comma is the first character on the line, the contents of the port at the address shown will not be modified and the display will advance to the next port. This feature may be used to examine and modify ports one at a time, skipping over ports that the user does not wish to change. If the last data value on a line is not terminated by a comma, the command terminates.

The maximum value that may be specified for the starting port number is 255. If, during the entry of data into the I/O ports, the maximum port number is exceeded, the command will terminate with the message "ADDRESS OUT OF RANGE". All data entered up to this point will have been placed into the port latches. The maximum value that may be specified for a data value is ØFFH.

These commands enable the user to specify the preset value to be used with a port when the port is declared preset with the IP or OP commands. Keep in mind that if an input port is preset (IP command), its value will not change except by use of the SIN or FIN command. However, any output instruction will change the value placed into an output port by the SOUT command.

Example:

```
     SIN   0 1
     SOUT  4  ØFH,2
```

Error Conditions:

1. No starting port number specified for command
2. Data value greater than 255
3. Port number assumes value greater than 255 during command
4. Invalid operand

<u>SIB</u> — Read Serial Input Data from Data Buffer
<u>SIC</u> — Read Serial Input Data from Current Input Device
<u>SIP</u> — Read Serial Input Data from Preset SID Latch
<u>SIS</u> — Read Serial Input Data from Standard Input Device

SIx

These four commands allow the user wide flexability in the simulation of Serial Input. The Serial Input Port may "read" its input data from any one of the sources described below. Of course, the source of a port's input data can be altered at any time during the simulation.

The Serial input commands, SIB, SIC, SIP, and SIS, are analogous to the Normal Port Input commands, IB, IC, IP, and IS. A brief description of the Serial Input Commands is given here. The user may refer to the Port Input Command descriptions for more details.

SIB - read serial input data from Data Buffer. Data is entered into the Data Buffer by the SDAT Command.

SIC - read serial input data from the current input device.

SIS - read serial input data from the standard input device. The following message is displayed at the standard input device in the interactive mode:

```
*PPPP    INPUT ON SERIAL PORT =
```

where PPPP is the address of the input instruction.

SIP - read serial input data from the value preset into the SID latch. The value may be preset by the SET command. Serial input is initially set to this mode.

Every time a RIM instruction is executed, an input data value is supplied. If the user is not interested in simulating serial I/O, he should leave the mode set to preset input, the default. When a RIM instruction is executed, no message will be issued asking for input data, and the user can ignore the SID bit.

The actions taken when an input data error is encountered are the same as those actions taken for normal input port errors. These error actions are discussed in the Normal Port I/O Command descriptions (IB, IC, IP, and IS).

Examples:

```
     SIC
     SIB
```

SM — Set Memory

```
SM     A   V {,V, V, ...}
```

This command is used to enter and/or examine data in the simulated Memory. The first operand of this command specifies a Memory address at which the following data will be entered. The first data byte (V) will be entered at the starting address, "A", and successive data bytes will be entered at successive addresses.

The user may continue this command on additional lines by terminating the last data on a line with a comma. If the command is continued, the following line will display the address of the next memory location followed by the contents of that location. For example:

```
Ø3A2  67 -
```

The user may then modify the contents of this location as well as enter as many data values as required starting at the address shown. If a comma is the first character on the line, the contents of memory at the address shown will not be modified and the display will advance to the next address. If the last data value on a line is not terminated by a comma, the command terminates.

The maximum value that may be specified for the starting address is ØFFFFH or the maximum memory size set at compile time if smaller. If, during the entry of data into the memory, the maximum memory size is exceeded, the command will terminate with the message "ADDRESS OUT OF RANGE". All data entered up to this point will have been placed into the memory. The maximum value that may be specified for a data byte is ØFFH.

Example: (simulator output is underlined)

```
SM   2ØØH  5, 3,ØB5H
SM   Ø  1,
ØØØ1 Ø9 - ,
ØØØ2 45 - 45,46
```

Error Conditions:

1. Starting address not specified
2. Data value greater than 255
3. Address assumes value larger than ØFFFFH during command
4. Invalid operand

SOC — Write Serial Output Data to Current Output Device
SOP — Write Serial Output Data to SOD Latch
SOS — Write Serial Output Data to Standard Output Device

SOx

These three commands allow the user wide flexability in the simulation of serial output. The Serial output port may "write" its data to one of the destinations described below. The destination of a serial port's output data may be changed at any time during the simulation.

The serial output commands, SOC, SOP, and SOS, are analogous to the Normal Port Output Commands, OC, OP, and OS. A brief description of the serial commands is given here. The user may refer to the Port Output Command descriptions for more details.

SOC - write output data to the current output device. The following message is displayed:

```
*PPPP     OUTPUT ON SERIAL PORT = V
```

where PPPP is the address of the instruction writing the output data to the port and V is the output data value.

SOS - write output data to the standard output device. The same message described in the SOC command description is displayed.

SOP - write output data to SOD latch only. Serial output is initially set to this mode.

Every time a SIM instruction is executed, an output data value is written to the specified device. If the user is not interested in simulating serial I/O, he should leave the mode set to latched output only, the default. No messages will be issued. When a SIM instruction is executed, no message will be issued specifying the output data and the user can ignore the SOD latch.

Example:

```
SOC
SOP
```

SSYM — Set Symbols

```
SSYM    symbol string=V {,symbol string=V, ...}
```

The command is used to change the value of a symbol already in the symbol table or to enter a new symbol and its value into the symbol table. If a symbol specified by this command is already in the table, its value will be set to that specified by this command. If the symbol is not already in the symbol table, it will be placed into the symbol table.

The symbol strings used in this command may not have a value placed after the last symbol, e.g. AB/CD+5. "V" may be any valid expression and may itself contain symbols. This includes the symbol actually being defined by this command if it already exists in the symbol table. The equal sign between the symbol and the value is optional and may be replaced by a blank.

Example:

```
SSYM START=5
SSYM DATA/ENTRY1=3,TABLE=1FH
```

Error Conditions:

1. Symbol table is full
2. Invalid symbol string format
3. Operand errcr

T — Trace Instructions
TA — Trace Instructions until Address
TB — Trace Instructions with Breakpoints

```
T       {N}
TA       A    {N}
TB      {N}
```

These commands cause the program to begin execution of instructions. The standard display line is displayed after each instruction has been executed.

The optional "N" parameter on the T and TB commands specifies the number of instructions that will be executed before the command terminates. This value overrides the one specified by the LI command.

The TA command is similar to the T command except that the program will continue execution until the address "A" is executed. "A" specifies an instruction address. If "N" is also specified, the instruction at address "A" will be executed "N" times before the command terminates.

The TB command is the same as the T command except execution will also terminate at an instruction breakpoint if one is encountered (see BP command).

These commands will also terminate execution under the following conditions:

1. Illegal instruction executed
2. Number of instructions specified in LI command executed

Example:

```
T    10
TA   177  5
TB   100
```

Error Conditions:

1. Address not specified for TA command

TR - Set Instruction Trace
NTR - Clear Instruction Trace

```
{N}TR  {R  {,R,  R, ...}}
```

The TR command enables the user to specify individual addresses or a range of addresses for which the standard display line will be printed during the "E" and "EA" command. Whenever the Simulator encounters an instruction address for which the trace flag has been set, the standard display line will be displayed. The format of the display will be that specified by the "FORM" command.

The NTR command enables the user to negate the effect of the TR command. Those addresses specified in the command will have their trace flag cleared so that no output occurs at the given address.

These commands may also be specified without any operands. In this case, the command effects only the master trace flag. When the NTR command is used without any operands, the master trace flag is turned off, inhibiting all checks for trace output during an "E" or "EA" command. However, the trace flags set by the TR command, if any, will remain set. Likewise, the TR command without operands turns the master trace flag back on. This feature is useful when the user wishes to execute a program without obtaining large amounts of output and then restore the trace information if that is desired.

Example:

```
TR     Ø ØFFH, 1FØH, 245
NTR
NTR    4, 6,67H
```

Error Conditions:

1. Invalid operand
2. Ending address less than starting address in range list

TYPE — Specify Processor Type

```
TYPE     {8080   8085}
```

This command is used to specify the microprocessor that is being simulated. If this command is not specified, it is assumed that the 8085 is being simulated.

When the 8080 is specified as the microprocessor being simulated, the RIM and SIM opcodes are detected as illegal opcodes. Also, 8080 instruction cycle counts are used instead of 8085 counts.

Example:

```
TYPE     8080
TYPE     8085
```

Error Conditions:

1. Illegal processor type specified.

W — Write Output

W   {*} {I/O device} {file name}

The W command is used to write subsequent simulator output to an alternate I/O device or file. This command is typically used when in the interactive mode to direct the results of the instruction execution to a line printer.

Output may be directed to the destinations listed below:

* - direct subsequent output to the standard output device that was specified in the program at compile time. This is typically a terminal in interactive mode or a line printer in batch mode. This operand would be used to return to normal operation after writing simulation results to a line printer or a disk file.

I/O device - direct subsequent output to the FORTRAN logical I/O device specified.

file name - direct subsequent output to the file specified. If the file does not exist, the message "FILE NOT FOUND" will be printed.

Any error conditions that occur in the interactive mode will be displayed at both the standard interactive output device as well as the device specified in the W command.

Example:

```
W    5
W    LIST
```

Error Conditions:

1. File not found
2. Invalid I/O device number specified

X - Exit Simulator

X

The X command is used to exit the simulator. Control is returned to the Host Computer's operating system.

CR — Single Step Execution

The Simulator has been designed to allow the user to trace one instruction without having to specify a complete command. Depressing a carriage return key with no other characters on the input line performs the same function as a "T 1" instruction. The ability to single step through the program by merely depressing the carriage return key is extremely useful. This allows the user to easily follow the program execution at his own pace.

This capability is dependent upon the ability of the program to detect an end-of-file (EOF) condition on a command input line (see instruction notes). On most computer systems, an EOF from an interactive device is indicated when the carriage return key is depressed with no other characters on the input line. For a batch device, an EOF is indicated when an attempt is made to read additional input data when none is present.

In the batch mode, if an EOF is detected while in the read mode (reading commands from a file), the Simulator will revert to the standard input mode. If an EOF is detected while reading commands from the standard input device, the program will terminate.

In the interactive mode, if an EOF is detected while in the read mode (reading commands from a file), the Simulator will revert to the standard input mode. If an EOF is detected while reading commands from the standard input device, one instruction will be traced as though a "T 1" command has been specified.

For both the batch and interactive modes, a blank line will also result in the tracing of the next instruction. This feature has been implemented to make the single step feature available even if the EOF condition cannot be detected.

SIMULATOR EXAMPLES

The following pages show the results of two simulation sessions. The first is a sample simulation showing the debugging of a binary to BCD conversion program. The second simulation is that of the test program supplied with the simulator. This program is used to verify the operation of the Simulator.

Sample Simulation

Figure 4-1 is an assembly listing of the Binary to BCD conversion subroutine along with a main program which calls the subroutine for testing purposes. The program was assembled using Microtec's 8080/8085 Macro Assembler. The object module output of the assembler was then placed onto the standard object module input device of the Simulator. Figure 4-2 shows the simulation session used to debug the program. The comments in the simulation describe the progress of the simulation session.

Test Program

The Simulator test program (object module and commands) is supplied with the Simulator and is used to verify the operation of the Simulator. Figure 4-3 shows the test program command stream and input object module. Figure 4-4 is the resulting output listing of the test program performed in the interactive mode.

To execute the test program, the user should perform the following steps:

Interactive Mode

1. Place the test program object module on the standard object module input device.
2. Enter the commands shown in Figure 4-3 interactively. Or
3. Place the test program command stream supplied with the program on an alternate command input device and enter a

RE command at the interactive device. This will cause the simulator to read and execute the commands in the read mode.

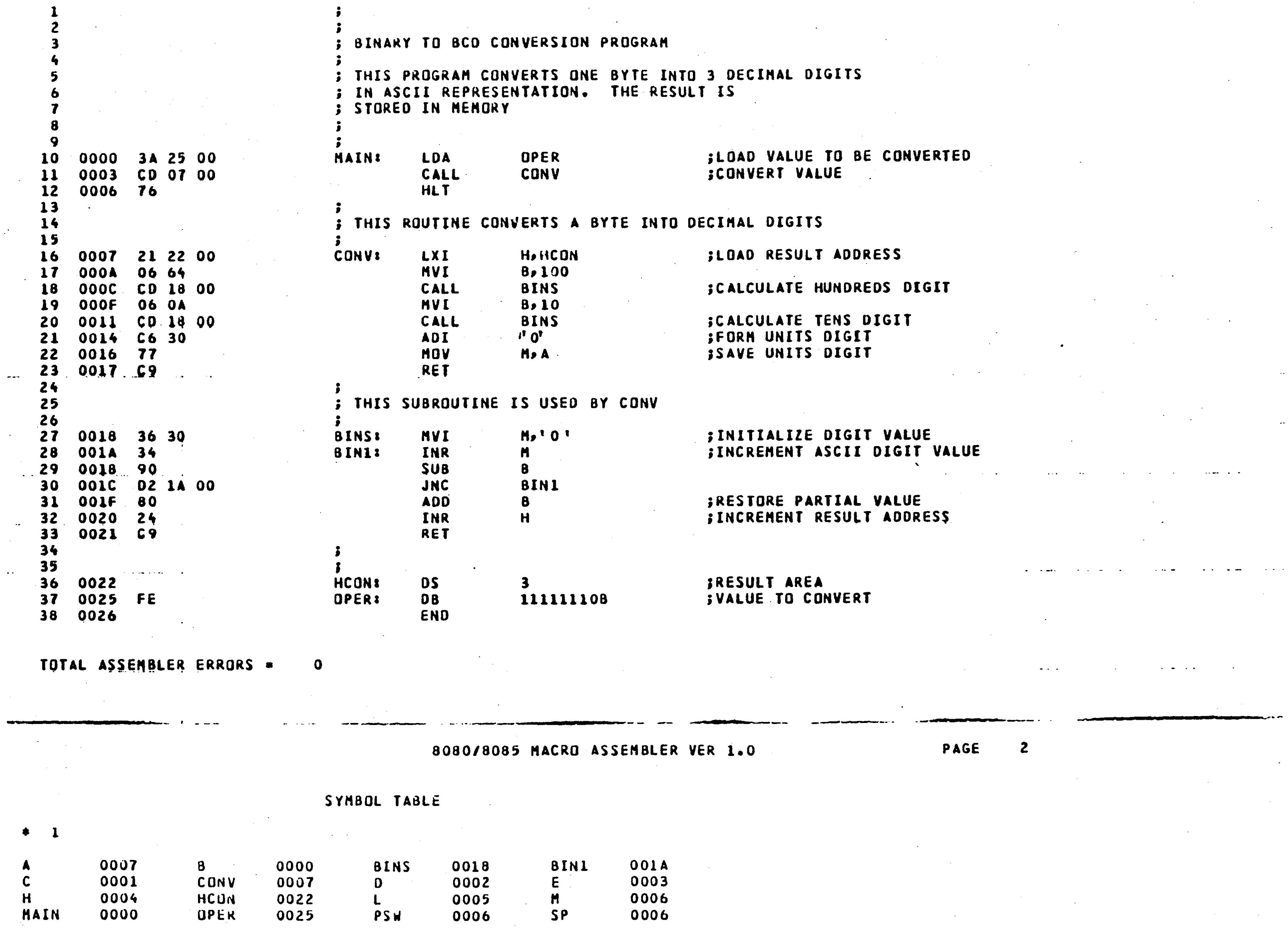
4. Examine the results of the simulation and compare them to the listing shown in Figure 4-4.

Batch Mode

1. Place the test program object module on the standard object module input device.
2. Place the test program command stream on the standard command input device.
3. Execute the program.
4. Examine the results and compare them to the listing shown in Figure 4-4.

If the user executes the test program in the batch mode or if he uses the RE command in the interactive mode, the object module and input commands may be read from the same input device. In this case, the object module should be placed into the command stream immediately after the "L *" command. Of course the device specified in the L command will have to be changed since the object module is not being read from the standard device but from the command device. Thus if the command input device is unit 5, the user would change the Load command as shown below.

```
RE      5           (used in interactive mode, step 3)
{test program commands}
L       5
{object module}
D       0   3FH
{remainder of commands}
```



```
  1                                        ;
  2                                        ;
  3                                        ; BINARY TO BCD CONVERSION PROGRAM
  4                                        ;
  5                                        ; THIS PROGRAM CONVERTS ONE BYTE INTO 3 DECIMAL DIGITS
  6                                        ; IN ASCII REPRESENTATION.  THE RESULT IS
  7                                        ; STORED IN MEMORY
  8                                        ;
  9                                        ;
 10  0000  3A 25 00                        MAIN:    LDA       OPER            ;LOAD VALUE TO BE CONVERTED
 11  0003  CD 07 00                                 CALL      CONV            ;CONVERT VALUE
 12  0006  76                                       HLT
 13                                        ;
 14                                        ; THIS ROUTINE CONVERTS A BYTE INTO DECIMAL DIGITS
 15                                        ;
 16  0007  21 22 00                        CONV:    LXI       H,HCON          ;LOAD RESULT ADDRESS
 17  000A  06 64                                    MVI       B,100
 18  000C  CD 18 00                                 CALL      BINS            ;CALCULATE HUNDREDS DIGIT
 19  000F  06 0A                                    MVI       B,10
 20  0011  CD 18 00                                 CALL      BINS            ;CALCULATE TENS DIGIT
 21  0014  C6 30                                    ADI       '0'             ;FORM UNITS DIGIT
 22  0016  77                                       MOV       M,A             ;SAVE UNITS DIGIT
 23  0017  C9                                       RET
 24                                        ;
 25                                        ; THIS SUBROUTINE IS USED BY CONV
 26                                        ;
 27  0018  36 30                           BINS:    MVI       M,'0'           ;INITIALIZE DIGIT VALUE
 28  001A  34                              BIN1:    INR       M               ;INCREMENT ASCII DIGIT VALUE
 29  001B  90                                       SUB       B
 30  001C  D2 1A 00                                 JNC       BIN1
 31  001F  80                                       ADD       B               ;RESTORE PARTIAL VALUE
 32  0020  24                                       INR       H               ;INCREMENT RESULT ADDRESS
 33  0021  C9                                       RET
 34                                        ;
 35                                        ;
 36  0022                                  HCON:    DS        3               ;RESULT AREA
 37  0025  FE                              OPER:    DB        11111110B       ;VALUE TO CONVERT
 38  0026                                           END

 TOTAL ASSEMBLER ERRORS =     0


                                                 8080/8085 MACRO ASSEMBLER VER 1.0                 PAGE     2


                                  SYMBOL TABLE

*  1

A        0007      B        0000      BINS     0018      BIN1     001A
C        0001      CONV     0007      D        0002      E        0003
H        0004      HCON     0022      L        0005      M        0006
MAIN     0000      OPER     0025      PSW      0006      SP       0006
```

Figure 4-1

```
 8080/8085 INTERACTIVE SIMULATOR VER 1.0

-*
-* SIMULATION RUN FOR BINARY TO ASCII BCD CONVERSION PROGRAM
-*
-* LOAD OBJECT CODE FROM THE STANDARD INPUT DEVICE
-*
-L  *
 *** NUMBER OF BYTES READ =   35
-*
-* THE PROGRAM INITIALLY HAD A DATA VALUE PLACED IN THE LOCATION HCON.
-* ASSUMING THE PROGRAM WORKS, THE VALUE IN HCON SHOULD BE CONVERTED
-* TO BCD DIGITS AND PLACED IN THE OPER ARRAY.
-*
-* EXAMINE THE BINARY VALUE TO BE CONVERTED AND THE RESULT AREA
-DM  22H 25H
0022  76 76 76 FE
-* EXECUTE THE PROGRAM LETTING THE PROGRAM STOP AT
-* THE HALT INSTRUCTION IN THE MAIN PROGRAM.
-* NOTE, THE PROGRAM COUNTER IS INITIALIZED TO ZERO BY THE SIMULATOR,
-* SO IT DOES NOT HAVE TO BE SET.  HOWEVER THE STACK POINTER SHOULD
-* BE INITIALIZED BEFORE IT IS USED.
-SET  SP=100H
-E

 PC       INST         EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0006  HLT                      0007  00000  34 0A 00 00 00 02 22  0100  0386
-* EXAMINE THE RESULTS AND THE BINARY VALUE TO BE CONVERTED
-DM  22H 25H
0022  33 76 76 FE
-* THE RESULTS SHOULD HAVE BEEN THE HEXADECIMAL NUMBERS  32,35,34 .
-* AS CAN BE SEEN ONLY THE FIRST BYTE IN THE OPER ARRAY WAS ALTERED.
-* THE PROGRAM MUST NOT BE PROPERLY INCREMENTING THE POINTER TO THE
-* RESULT AREA.  IN EXAMING THE PROGRAM IT CAN BE SEEN THAT THE
-* WRONG MNEMONIC WAS ENCODED FOR THE INCREMENT INSTRUCTION.
-* THE MNEMONIC SHOULD HAVE BEEN INX WHICH INCREMENTS THE HL PAIR
-* AND NOT INR WHICH ONLY INCREMENTS THE H REGISTER.
-* PATCH THE INCORRECT INSTRUCTION
-SM  20H  23H
-* USE EXECUTE COMMAND TO RUN UNTIL THE INCREMENT INSTRUCTION IS EXECUTED
-SET  PC=0
-EA  20H
0020  INX  H                   0021  10010  36 64 00 00 00 00 23  00FC  0541
-* AS CAN BE SEEN, THE POINTER IS INCREMENTING CORRECTLY NOW.
-* RUN PROGRAM UNTIL POINTER IS INCREMENTED A SECOND TIME
-EA  20H
0020  INX  H                   0021  10001  04 0A 00 00 00 00 24  00FC  0737
-* RUN TO COMPLETION
-E
0006  HLT                      0007  00000  34 0A 00 00 00 00 24  0100  0776
-* EXAMINE RESULTS AGAIN
-DM  22H 25H
0022  33 36 34 FE
-* THE RESULT IS STILL WRONG.  THE FIRST TWO DIGITS ARE OFF BY
-* A COUNT OF ONE.
-* RESET THE PROGRAM COUNTER AND TRACE THE PROGRAM FLOW FOR FIRST DIGIT
-SET  PC=0
-TA  0FH
0000  LDA  0025      0025 FE   0003  00000  FE 0A 00 00 00 00 24  0100  0789
0003  CALL 0007                0007  00000  FE 0A 00 00 00 00 24  00FE  0807
0007  LXI  H,0022              000A  00000  FE 0A 00 00 00 00 22  00FE  0817
000A  MVI  B,64                000C  00000  FE 64 00 00 00 00 22  00FE  0824
```

Figure 4-2

```
0018  SUB  B                 001C  00111  9A 64 00 00 00 00 22  00FC  0866
001C  JNC  001A              001A  00111  9A 64 00 00 00 00 22  00FC  0876
001A  INR  M       0022 32   001B  00000  9A 64 00 00 00 00 22  00FC  0886
001B  SUB  B                 001C  00011  36 64 00 00 00 00 22  00FC  0890
001C  JNC  001A              001A  00011  36 64 00 00 00 00 22  00FC  0900
001A  INR  M       0022 33   001B  00010  36 64 00 00 00 00 22  00FC  0910
001B  SUB  B                 001C  10111  D2 64 00 00 00 00 22  00FC  0914
001C  JNC  001A              001F  10111  D2 64 00 00 00 00 22  00FC  0921
001F  ADD  B                 0020  10010  36 64 00 00 00 00 22  00FC  0925

 PC      INST         EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0020  INX  H                 0021  10010  36 64 00 00 00 00 23  00FC  0931
0021  RET                    000F  10010  36 64 00 00 00 00 23  00FE  0941
000F  MVI  B,0A              0011  10010  36 0A 00 00 00 00 23  00FE  0948
-* IN EXAMING THE PROGRAM FLOW, IT CAN BE SEEN THE DIGIT IN MEMORY
-* IS BEING INCREMENTED ONE TIME MORE THAN NECESSARY.
-* THIS COULD BE FIXED BY CHANGING THE INITIALIZED VALUE FROM
-* AN ASCII  0  TO AN ASCII  0 -1  .
-* CHANGE THIS VALUE AND TRY AGAIN
-SM  19H  2FH
-RES
-SET SP=100H
-E
0006  HLT                    0007  00000  34 0A 00 00 00 00 24  0100  0390
-* EXAMINE RESULTS
-DM  22H 25H
0022  32 35 34 FE
-*
-* THE RESULT IN NOW CORRECT.  THE PROGRAM HAS BEEN DEBUGGED.
-*
-X
```

Figure 4-2

```
*
* LOAD OBJECT MODULE FROM STANDARD DEVICE
L *
* CHECK DISPLAY MEMORY, ALSO CHECK VARIOUS NUMBER BASE DESCRIPTORS
DM 1010B,0AH,10,12Q
D  0 3FH
BASE H
DM 10
BASE  D
* DISPLAY HEADING AND CPU STATUS
H   0
DC
* DISPLAY INTERRUPT STATUS AND SERIAL I/O
DIM
* TURN ON THE SHORT FORMAT OF THE STANDARD DISPLAY LINE
FORM S
T 1
DC
FORM L
DC
T 1
* RESET MICROPROCESSOR
RES
DC
* TEST MOVE IMMEDIATE AND REGISTER MOVE INSTRUCTIONS
TA  0CH
SET  A=0
TA  14H
SET  B=12H
TA  1BH
SET  C=23H
TA  22H
SET  D=34H
TA  29H
SET  E=45H
TA  30H
SET  H=56H
TA  37H
SET  L=67H
TA  3EH
BP  57H
TB
DM 5600H 560FH
TA  69H
DM 5600H 560FH
TA  8BH
DM  5800H 580FH
TA  93H
DM  5640H 5650H
T  4
SET SP=564DH
T  3
* TEST ARITHMETIC INSTRUCTIONS
TA  111H
DM  5800H 580FH
TA  163H
* TEST JUMP INSTRUCTIONS
SET  CY=0,P=0,Z=0,S=0
TA  188H
SET  CY=1,P=1,Z=1,S=1
TA  184H
```

Figure 4-3

```
SET  CY=0,P=0,Z=0,S=0
TA  201H
SET  CY=1,P=1,Z=1,S=1
TA  219H
T  2
DH
* TEST NORMAL PORT I/O INSTRUCTIONS
SET PC=21AH
T  6
23H
45H
DI  2,0F0H
DO  5,0E0H
IP  2,0F0H
OP  5,0E0H
T  5
DI  2,0F0H
DO  5,0E0H
* TEST MEMORY MAPPED I/O INSTRUCTIONS
MIC  5A00H
MOC  5A00H
T  5
0A6H
0BFH
DM  5A00H
MIB  5A00H
MOP  5A00H
MDAT  5A00H 98H 0E1H
SET  PC=230H
T  5
DM  5A00H
* TEST SERIAL I/O
DIM
T  3
DIM
SET  SI=1
DIM
T  1
SIB
SDAT  0 1 1 0
T  5
SET  PC=240H
T  5
SIC
SOC
T  5
1
* TEST VARIOUS INTERRUPTS
INT  I  7  0C3H 74H 01H
SET  IE=1
T  4
SET PC=24CH
INT T 2
T  4
SET PC=24CH
* THIS INTERRUPT WILL NOT BE RECOGNIZED SINCE ENABLE BIT IS RESET
INT  I 2 0
T  4
SET PC=250H
SET  IM=0
DIM
```

Figure 4-3

```
SET PC=25CH
SET IE=1
SET IM=1
* THIS INTERRUPT WILL NOT BE RECOGNIZED BECAUSE IT IS MASKED
INT  5  0
T  1
DIM
* INTERRUPT 7.5 WILL NOT BE RECOGNIZED UNTIL INT7.5 FLIPFLOP IS RESET
INT  7  0
T  7
* SET SOME SYMBOL VALUES
SSYM BEGIN=52, START=25H, STOP=100H
DSYM
DSYM BEGIN
DM   START  BEGIN
DEL BEGIN
DSYM
SM  0FFF2H 25H
DM  0FFF0H 0FFFFH
X
```

Figure 4-3

```
:1C0038007D4540555D656D215150113130017170314140 7E2346234E2356235EFF
:1C005400236623 6E21005677237023712372237323742375233646015650115759
:1C007000500A1A3A58503E5611025801055802 3E25123EFE3200582A5050220692
:1C008C0058315056C5D5E5F5E1C1F1D1EBE3F906050EA016FF1E5026202E028751
:1C00A800808182838485C601C6FF8F88898A8B8C8DCE559F98999A9B9C9DDE0193
:1C00C400DEFF97909192939495D655A73EA5A0A13E5AA2A3A43EFFA5E655B78042
:1C00E000B1B23E00B3B4B5F655AFA8A9AAABACADEEAABFB8B9BABBBCBDFE502129
:1C00FC005150868E9E96A6B6AEBE2100573C040C141C242C34343D050D151D25E5
:1C0118002D35353E22C64427C688272F373F3F01101011A55A091929390313235C
:1C013400330B1B2B3B3E5A07070707070707070F0F0F0F0F0F0F0F171717171735
:1C015000171717171F1F1F1F1F1F1F1FC3620176000000DA6001CA6001EA60D4
:1C016C0001FA6001D27601C36001C27C01C36001E28201C36001F28801C3600123
:1C01880000D26001C26001E26001F26001DA9B01C36001CAA101C36001EAA701B3
:1C01A400C36001FAAD01C3600121B401E9C36001CD5B50CC5D50DC5D50EC5D5059
:1C01C000FC5D50C45B50D45B50E45B50F45B5000CC5B50DC5B50EC5B50FC5B5022
:1C01DC00C45D50D45D50E45D50F45D5000CD5E50CD6050CD6250CD6450CD665068
:1C01F800CD6850CD6A50CD6C50CD5E50CD6050CD6250CD6450CD6650CD6850CD89
:1C0214006A50CD6C50C7DB02D8F03E28D3053E96D3E0DB02D8F0D3053E87D3E05F
:1C0230003A005A21005A7E32005A73203ECF30202020202020203EC0303E8030CD
:18024C00785000760E511665F87C0650790080177B01060517F37D0097
:1C5050002223334452627785DC5A5A78C976C8C9D8C9E8C9F8C9C0C9D0C9E0C922
:02506C00F0C989
:00000001FF
```

Figure 4-3

```
 8080/8085 INTERACTIVE SIMULATOR VER 1.0

-*
-*     *** 8080/8085 SIMULATOR TEST DECK
-*
-* LOAD OBJECT MODULE FROM STANDARD DEVICE
-L *
 *** NUMBER OF BYTES READ =  642
-* CHECK DISPLAY MEMORY, ALSO CHECK VARIOUS NUMBER BASE DESCRIPTORS
-DM 1010B,0AH,10,12Q
000A  26
000A  26
000A  26
000A  26
-D  0 3FH
0000  3E 01 06 02   0E 03 16 04   1E 50 26 88   2E FF 7F 47
0010  4F 57 5F 67   6F 78 40 48   50 58 60 68   79 41 49 51
0020  59 61 69 7A   42 4A 52 5A   62 6A 7B 43   4B 53 5B 63
0030  6B 7C 44 4C   54 5C 64 6C   7D 45 4D 55   5D 65 6D 21
-BASE H
-DM 10
0010  4F
-BASE  D
-* DISPLAY HEADING AND CPU STATUS
-H   0

 PC       INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP    CYC
-DC
0000                  0010 4F 0000  00000  00 00 00 00 00 00 00  0000  0000
-* DISPLAY INTERRUPT STATUS AND SERIAL I/O
-DIM
    IM = 00000000   SOD = 0  INT7.5 = 0
-* TURN ON THE SHORT FORMAT OF THE STANDARD DISPLAY LINE
-FORM S
-T 1
0000  MVI  A,01       01
-DC
0000  MVI  A,01               0002  00000  01 00 00 00 00 00 00  0000  0007
-FORM L
-DC
0000  MVI  A,01               0002  00000  01 00 00 00 00 00 00  0000  0007
-T 1

 PC       INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP    CYC
0002  MVI  B,02               0004  00000  01 02 00 00 00 00 00  0000  0014
-* RESET MICROPROCESSOR
-RES
-DC
0002  MVI  B,02               0000  00000  00 00 00 00 00 00 00  0000  0000
-* TEST MOVE IMMEDIATE AND REGISTER MOVE INSTRUCTIONS
-TA  0CH
0000  MVI  A,01               0002  00000  01 00 00 00 00 00 00  0000  0007
0002  MVI  B,02               0004  00000  01 02 00 00 00 00 00  0000  0014
0004  MVI  C,03               0006  00000  01 02 03 00 00 00 00  0000  0021
0006  MVI  D,04               0008  00000  01 02 03 04 00 00 00  0000  0028
0008  MVI  E,50               000A  00000  01 02 03 04 50 00 00  0000  0035
000A  MVI  H,88               000C  00000  01 02 03 04 50 88 00  0000  0042
000C  MVI  L,FF               000E  00000  01 02 03 04 50 88 FF  0000  0049
-SET  A=0
-TA  14H
000E  MOV  A,A                000F  00000  00 02 03 04 50 88 FF  0000  0053
000F  MOV  B,A                0010  00000  00 00 03 04 50 88 FF  0000  0057

 PC       INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP    CYC
```

Figure 4-4

```
0010  MOV  C,A               0011  00000  00 00 00 04 50 88 FF  0000  0061
0011  MOV  D,A               0012  00000  00 00 00 00 50 88 FF  0000  0065
0012  MOV  E,A               0013  00000  00 00 00 00 00 88 FF  0000  0069
0013  MOV  H,A               0014  00000  00 00 00 00 00 00 FF  0000  0073
0014  MOV  L,A               0015  00000  00 00 00 00 00 00 00  0000  0077
-SET  B=12H
-TA  18H
0015  MOV  A,B               0016  00000  12 12 00 00 00 00 00  0000  0081
0016  MOV  B,B               0017  00000  12 12 00 00 00 00 00  0000  0085
0017  MOV  C,B               0018  00000  12 12 12 00 00 00 00  0000  0089
0018  MOV  D,B               0019  00000  12 12 12 12 00 00 00  0000  0093
0019  MOV  E,B               001A  00000  12 12 12 12 12 00 00  0000  0097

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP    CYC
001A  MOV  H,B               001B  00000  12 12 12 12 12 12 00  0000  0101
001B  MOV  L,B               001C  00000  12 12 12 12 12 12 12  0000  0105
-SET  C=23H
-TA  22H
001C  MOV  A,C               001D  00000  23 12 23 12 12 12 12  0000  0109
001D  MOV  B,C               001E  00000  23 23 23 12 12 12 12  0000  0113
001E  MOV  C,C               001F  00000  23 23 23 12 12 12 12  0000  0117
001F  MOV  D,C               0020  00000  23 23 23 23 12 12 12  0000  0121
0020  MOV  E,C               0021  00000  23 23 23 23 23 12 12  0000  0125
0021  MOV  H,C               0022  00000  23 23 23 23 23 23 12  0000  0129
0022  MOV  L,C               0023  00000  23 23 23 23 23 23 23  0000  0133
-SET  D=34H
-TA  29H
0023  MOV  A,D               0024  00000  34 23 23 34 23 23 23  0000  0137

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP    CYC
0024  MOV  B,D               0025  00000  34 34 23 34 23 23 23  0000  0141
0025  MOV  C,D               0026  00000  34 34 34 34 23 23 23  0000  0145
0026  MOV  D,D               0027  00000  34 34 34 34 23 23 23  0000  0149
0027  MOV  E,D               0028  00000  34 34 34 34 34 23 23  0000  0153
0028  MOV  H,D               0029  00000  34 34 34 34 34 34 23  0000  0157
0029  MOV  L,D               002A  00000  34 34 34 34 34 34 34  0000  0161
-SET  E=45H
-TA  30H
002A  MOV  A,E               002B  00000  45 34 34 34 45 34 34  0000  0165
002B  MOV  B,E               002C  00000  45 45 34 34 45 34 34  0000  0169
002C  MOV  C,E               002D  00000  45 45 45 34 45 34 34  0000  0173
002D  MOV  D,E               002E  00000  45 45 45 45 45 34 34  0000  0177

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP    CYC
002E  MOV  E,E               002F  00000  45 45 45 45 45 34 34  0000  0181
002F  MOV  H,E               0030  00000  45 45 45 45 45 45 34  0000  0185
0030  MOV  L,E               0031  00000  45 45 45 45 45 45 45  0000  0189
-SET  H=56H
-TA  37H
0031  MOV  A,H               0032  00000  56 45 45 45 45 56 45  0000  0193
0032  MOV  B,H               0033  00000  56 56 45 45 45 56 45  0000  0197
0033  MOV  C,H               0034  00000  56 56 56 45 45 56 45  0000  0201
0034  MOV  D,H               0035  00000  56 56 56 56 45 56 45  0000  0205
0035  MOV  E,H               0036  00000  56 56 56 56 56 56 45  0000  0209
0036  MOV  H,H               0037  00000  56 56 56 56 56 56 45  0000  0213
0037  MOV  L,H               0038  00000  56 56 56 56 56 56 56  0000  0217
-SET  L=67H
-TA  3EH

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP    CYC
0038  MOV  A,L               0039  00000  67 56 56 56 56 56 67  0000  0221
0039  MOV  B,L               003A  00000  67 67 56 56 56 56 67  0000  0225
003A  MOV  C,L               003B  00000  67 67 67 56 56 56 67  0000  0229
003B  MOV  D,L               003C  00000  67 67 67 67 56 56 67  0000  0233
003C  MOV  E,L               003D  00000  67 67 67 67 67 56 67  0000  0237
003D  MOV  H,L               003E  00000  67 67 67 67 67 67 67  0000  0241
```

```
003E  MOV  L,L              003F  00000  67 67 67 67 67 67 67  0000  0245
-BP  57H
-T8
003F  LXI  H,5051           0042  00000  67 67 67 67 67 50 51  0000  0255
0042  LXI  D,3031           0045  00000  67 67 67 30 31 50 51  0000  0265
0045  LXI  B,7071           0048  00000  67 70 71 30 31 50 51  0000  0275

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0048  LXI  SP,4041          004B  00000  67 70 71 30 31 50 51  4041  0285
004B  MOV  A,M      5051 23 004C  00000  23 70 71 30 31 50 51  4041  0292
004C  INX  H                004D  00000  23 70 71 30 31 50 52  4041  0298
004D  MOV  B,M      5052 33 004E  00000  23 33 71 30 31 50 52  4041  0305
004E  INX  H                004F  00000  23 33 71 30 31 50 53  4041  0311
004F  MOV  C,M      5053 44 0050  00000  23 33 44 30 31 50 53  4041  0318
0050  INX  H                0051  00000  23 33 44 30 31 50 54  4041  0324
0051  MOV  D,M      5054 52 0052  00000  23 33 44 52 31 50 54  4041  0331
0052  INX  H                0053  00000  23 33 44 52 31 50 55  4041  0337
0053  MOV  E,M      5055 62 0054  00000  23 33 44 52 62 50 55  4041  0344

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0054  INX  H                0055  00000  23 33 44 52 62 50 56  4041  0350
0055  MOV  H,M      5056 77 0056  00000  23 33 44 52 62 77 56  4041  0357
0056  INX  H                0057  00000  23 33 44 52 62 77 57  4041  0363
0057  MOV  L,M      7757 76 0058  00000  23 33 44 52 62 77 76  4041  0370
-DM 5600H 560FH
5600  76 76 76 76   76 76 76 76   76 76 76 76   76 76 76 76
-TA  69H
0058  LXI  H,5600           005B  00000  23 33 44 52 62 56 00  4041  0380
005B  MOV  M,A      5600 23 005C  00000  23 33 44 52 62 56 00  4041  0387
005C  INX  H                005D  00000  23 33 44 52 62 56 01  4041  0393
005D  MOV  M,B      5601 33 005E  00000  23 33 44 52 62 56 01  4041  0400
005E  INX  H                005F  00000  23 33 44 52 62 56 02  4041  0406
005F  MOV  M,C      5602 44 0060  00000  23 33 44 52 62 56 02  4041  0413

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0060  INX  H                0061  00000  23 33 44 52 62 56 03  4041  0419
0061  MOV  M,D      5603 52 0062  00000  23 33 44 52 62 56 03  4041  0426
0062  INX  H                0063  00000  23 33 44 52 62 56 04  4041  0432
0063  MOV  M,E      5604 62 0064  00000  23 33 44 52 62 56 04  4041  0439
0064  INX  H                0065  00000  23 33 44 52 62 56 05  4041  0445
0065  MOV  M,H      5605 56 0066  00000  23 33 44 52 62 56 05  4041  0452
0066  INX  H                0067  00000  23 33 44 52 62 56 06  4041  0458
0067  MOV  M,L      5606 06 0068  00000  23 33 44 52 62 56 06  4041  0465
0068  INX  H                0069  00000  23 33 44 52 62 56 07  4041  0471
0069  MVI  M,46     5607 46 006B  00000  23 33 44 52 62 56 07  4041  0481
-DM 5600H 560FH
5600  23 33 44 52   62 56 06 46   76 76 76 76   76 76 76 76
-TA  8BH

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
006B  LXI  B,5056           006E  00000  23 50 56 52 62 56 07  4041  0491
006E  LXI  D,5057           0071  00000  23 50 56 50 57 56 07  4041  0501
0071  LDAX B        5056 77 0072  00000  77 50 56 50 57 56 07  4041  0508
0072  LDAX D        5057 85 0073  00000  85 50 56 50 57 56 07  4041  0515
0073  LDA  5058     5058 DC 0076  00000  DC 50 56 50 57 56 07  4041  0528
0076  MVI  A,56             0078  00000  56 50 56 50 57 56 07  4041  0535
0078  LXI  D,5802           007B  00000  56 50 56 58 02 56 07  4041  0545
007B  LXI  B,5805           007E  00000  56 58 05 58 02 56 07  4041  0555
007E  STAX B        5805 56 007F  00000  56 58 05 58 02 56 07  4041  0562
007F  MVI  A,25             0081  00000  25 58 05 58 02 56 07  4041  0569

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0081  STAX D        5802 25 0082  00000  25 58 05 58 02 56 07  4041  0576
0082  MVI  A,FE             0084  00000  FE 58 05 58 02 56 07  4041  0583
0084  STA  5800     5800 FE 0087  00000  FE 58 05 58 02 56 07  4041  0596
0087  LHLD 5050     5051 23 008A  00000  FE 58 05 58 02 23 22  4041  0612
```

```
008.  SHLD 5806       5807 23  008D  00000  FE 58 05 58 02 23 22  4041  0628
-DM  5800H 580FH
5800  FE 76 25 76   76 56 22 23   76 76 76 76   76 76 76 76
-TA  93H
008D  LXI  SP,5650             0090  00000  FE 58 05 58 02 23 22  5650  0638
0090  PUSH B          .564E 05  0091  00000  FE 58 05 58 02 23 22  564E  0650
0091  PUSH D          564C 02  0092  00000  FE 58 05 58 02 23 22  564C  0662
0092  PUSH H          564A 22  0093  00000  FE 58 05 58 02 23 22  564A  0674
0093  PUSH PSW        5648 02  0094  00000  FE 58 05 58 02 23 22  5648  0686
-DM  5640H 5650H
5640  76 76 76 76   76 76 76 76   02 FE 22 23   02 58 05 58
5650  76
-T  4

 PC       INST         EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP    CYC
0094  POP  H          5649 FE  0095  00000  FE 58 05 58 02 FE 02  564A  0696
0095  POP  B          564B 23  0096  00000  FE 23 22 58 02 FE 02  564C  0706
0096  POP  PSW        564D 58  0097  00000  58 23 22 58 02 FE 02  564E  0716
0097  POP  D          564F 58  0098  00000  58 23 22 58 05 FE 02  5650  0726
-SET SP=564DH
-T  3
0098  XCHG                     0099  00000  58 23 22 FE 02 58 05  564D  0734
0099  XTHL            564E 58  009A  00000  58 23 22 FE 02 05 58  564D  0750
009A  SPHL                     009B  00000  58 23 22 FE 02 05 58  0558  0756
-* TEST ARITHMETIC INSTRUCTIONS
-TA  111H
009B  MVI  B,05                009D  00000  58 05 22 FE 02 05 58  0558  0763
009D  MVI  C,A0                009F  00000  58 05 A0 FE 02 05 58  0558  0770
009F  MVI  D,FF                00A1  00000  58 05 A0 FF 02 05 58  0558  0777

 PC       INST         EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP    CYC
00A1  MVI  E,50                00A3  00000  58 05 A0 FF 50 05 58  0558  0784
00A3  MVI  H,20                00A5  00000  58 05 A0 FF 50 20 58  0558  0791
00A5  MVI  L,02                00A7  00000  58 05 A0 FF 50 20 02  0558  0798
00A7  ADD  A                   00A8  00101  B0 05 A0 FF 50 20 02  0558  0802
00A8  ADD  B                   00A9  00100  B5 05 A0 FF 50 20 02  0558  0806
00A9  ADD  C                   00AA  10010  55 05 A0 FF 50 20 02  0558  0810
00AA  ADD  D                   00AB  10001  54 05 A0 FF 50 20 02  0558  0814
00AB  ADD  E                   00AC  00100  A4 05 A0 FF 50 20 02  0558  0818
00AC  ADD  H                   00AD  00100  C4 05 A0 FF 50 20 02  0558  0822
00AD  ADD  L                   00AE  00110  C6 05 A0 FF 50 20 02  0558  0826

 PC       INST         EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP    CYC
00AE  ADI  01                  00B0  00100  C7 05 A0 FF 50 20 02  0558  0833
00B0  ADI  FF                  00B2  10111  C6 05 A0 FF 50 20 02  0558  0840
00B2  ADC  A                   00B3  10110  8D 05 A0 FF 50 20 02  0558  0844
00B3  ADC  B                   00B4  00111  93 05 A0 FF 50 20 02  0558  0848
00B4  ADC  C                   00B5  10010  33 05 A0 FF 50 20 02  0558  0852
00B5  ADC  D                   00B6  10011  33 05 A0 FF 50 20 02  0558  0856
00B6  ADC  E                   00B7  00110  84 05 A0 FF 50 20 02  0558  0860
00B7  ADC  H                   00B8  00100  A4 05 A0 FF 50 20 02  0558  0864
00B8  ADC  L                   00B9  00110  A6 05 A0 FF 50 20 02  0558  0868
00B9  ACI  55                  00BB  00100  FB 05 A0 FF 50 20 02  0558  0875

 PC       INST         EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP    CYC
00BB  SBB  A                   00BC  01011  00 05 A0 FF 50 20 02  0558  0879
00BC  SBB  B                   00BD  10100  FB 05 A0 FF 50 20 02  0558  0883
00BD  SBB  C                   00BE  00011  5A 05 A0 FF 50 20 02  0558  0887
00BE  SBB  D                   00BF  10000  5B 05 A0 FF 50 20 02  0558  0891
00BF  SBB  E                   00C0  00011  0A 05 A0 FF 50 20 02  0558  0895
00C0  SBB  H                   00C1  10101  EA 05 A0 FF 50 20 02  0558  0899
00C1  SBB  L                   00C2  00111  E7 05 A0 FF 50 20 02  0558  0903
00C2  SBI  01                  00C4  00101  E6 05 A0 FF 50 20 02  0558  0910
00C4  SBI  FF                  00C6  10110  E7 05 A0 FF 50 20 02  0558  0917
00C6  SUB  A                   00C7  01011  00 05 A0 FF 50 20 02  0558  0921
```

```
 PC       INST          EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
00C7  SUB  B                    00C8  10100  F8 05 A0 FF 50 20 02  0558  0925
00C8  SUB  C                    00C9  00001  58 05 A0 FF 50 20 02  0558  0929
00C9  SUB  D                    00CA  10010  5C 05 A0 FF 50 20 02  0558  0933
00CA  SUB  E                    00CB  00011  0C 05 A0 FF 50 20 02  0558  0937
00CB  SUB  H                    00CC  10101  EC 05 A0 FF 50 20 02  0558  0941
00CC  SUB  L                    00CD  00101  EA 05 A0 FF 50 20 02  0558  0945
00CD  SUI  55                   00CF  00111  95 05 A0 FF 50 20 02  0558  0952
00CF  ANA  A                    00D0  00111  95 05 A0 FF 50 20 02  0558  0956
00D0  MVI  A,A5                 00D2  00111  A5 05 A0 FF 50 20 02  0558  0963
00D2  ANA  B                    00D3  00011  05 05 A0 FF 50 20 02  0558  0967

 PC       INST          EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
00D3  ANA  C                    00D4  01011  00 05 A0 FF 50 20 02  0558  0971
00D4  MVI  A,5A                 00D6  01011  5A 05 A0 FF 50 20 02  0558  0978
00D6  ANA  D                    00D7  00011  5A 05 A0 FF 50 20 02  0558  0982
00D7  ANA  E                    00D8  00011  50 05 A0 FF 50 20 02  0558  0986
00D8  ANA  H                    00D9  01011  00 05 A0 FF 50 20 02  0558  0990
00D9  MVI  A,FF                 00DB  01011  FF 05 A0 FF 50 20 02  0558  0997
00DB  ANA  L                    00DC  00001  02 05 A0 FF 50 20 02  0558  1001
00DC  ANI  55                   00DE  01011  00 05 A0 FF 50 20 02  0558  1008
00DE  ORA  A                    00DF  01010  00 05 A0 FF 50 20 02  0558  1012
00DF  ORA  B                    00E0  00010  05 05 A0 FF 50 20 02  0558  1016

 PC       INST          EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
00E0  ORA  C                    00E1  00110  A5 05 A0 FF 50 20 02  0558  1020
00E1  ORA  D                    00E2  00110  FF 05 A0 FF 50 20 02  0558  1024
00E2  MVI  A,00                 00E4  00110  00 05 A0 FF 50 20 02  0558  1031
00E4  ORA  E                    00E5  00010  50 05 A0 FF 50 20 02  0558  1035
00E5  ORA  H                    00E6  00000  70 05 A0 FF 50 20 02  0558  1039
00E6  ORA  L                    00E7  00010  72 05 A0 FF 50 20 02  0558  1043
00E7  ORI  55                   00E9  00010  77 05 A0 FF 50 20 02  0558  1050
00E9  XRA  A                    00EA  01010  00 05 A0 FF 50 20 02  0558  1054
00EA  XRA  B                    00EB  00010  05 05 A0 FF 50 20 02  0558  1058
00EB  XRA  C                    00EC  00110  A5 05 A0 FF 50 20 02  0558  1062

 PC       INST          EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
00EC  XRA  D                    00ED  00010  5A 05 A0 FF 50 20 02  0558  1066
00ED  XRA  E                    00EE  00010  0A 05 A0 FF 50 20 02  0558  1070
00EE  XRA  H                    00EF  00000  2A 05 A0 FF 50 20 02  0558  1074
00EF  XRA  L                    00F0  00010  28 05 A0 FF 50 20 02  0558  1078
00F0  XRI  AA                   00F2  00110  82 05 A0 FF 50 20 02  0558  1085
00F2  CMP  A                    00F3  01011  82 05 A0 FF 50 20 02  0558  1089
00F3  CMP  B                    00F4  00010  82 05 A0 FF 50 20 02  0558  1093
00F4  CMP  C                    00F5  10111  82 05 A0 FF 50 20 02  0558  1097
00F5  CMP  D                    00F6  10100  82 05 A0 FF 50 20 02  0558  1101
00F6  CMP  E                    00F7  00001  82 05 A0 FF 50 20 02  0558  1105

 PC       INST          EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
00F7  CMP  H                    00F8  00001  82 05 A0 FF 50 20 02  0558  1109
00F8  CMP  L                    00F9  00101  82 05 A0 FF 50 20 02  0558  1113
00F9  CPI  50                   00FB  00001  82 05 A0 FF 50 20 02  0558  1120
00FB  LXI  H,5051               00FE  00001  82 05 A0 FF 50 50 51  0558  1130
00FE  ADD  M            5051 23 00FF  00110  A5 05 A0 FF 50 50 51  0558  1137
00FF  ADC  M            5051 23 0100  00100  C8 05 A0 FF 50 50 51  0558  1144
0100  SBB  M            5051 23 0101  00111  A5 05 A0 FF 50 50 51  0558  1151
0101  SUB  M            5051 23 0102  00111  82 05 A0 FF 50 50 51  0558  1158
0102  ANA  M            5051 23 0103  00001  02 05 A0 FF 50 50 51  0558  1165
0103  ORA  M            5051 23 0104  00000  23 05 A0 FF 50 50 51  0558  1172

 PC       INST          EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0104  XRA  M            5051 23 0105  01010  00 05 A0 FF 50 50 51  0558  1179
0105  CMP  M            5051 23 0106  10110  00 05 A0 FF 50 50 51  0558  1186
0106  LXI  H,5700               0109  10110  00 05 A0 FF 50 57 00  0558  1196
0109  INR  A                    010A  10000  01 05 A0 FF 50 57 00  0558  1200
010A  INR  B                    010B  10010  01 06 A0 FF 50 57 00  0558  1204
```

```
0108  INR  C                  010C  10100  01 06 A1 FF 50 57 00  0558  1208
010C  INR  D                  010D  11011  01 06 A1 00 50 57 00  0558  1212
010D  INR  E                  010E  10000  01 06 A1 00 51 57 00  0558  1216
010E  INR  H                  010F  10000  01 06 A1 00 51 58 00  0558  1220
010F  INR  L                  0110  10000  01 06 A1 00 51 58 01  0558  1224

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0110  INR  M       5801 77  0111  10010  01 06 A1 00 51 58 01  0558  1234
0111  INR  M       5801 78  0112  10010  01 06 A1 00 51 58 01  0558  1244
-DM  5800H 580FH
5800  FE 78 25 76   76 56 22 23   76 76 76 76   76 76 76 76
-TA  163H
0112  DCR  A                  0113  11011  00 06 A1 00 51 58 01  0558  1248
0113  DCR  B                  0114  10011  00 05 A1 00 51 58 01  0558  1252
0114  DCR  C                  0115  10111  00 05 A0 00 51 58 01  0558  1256
0115  DCR  D                  0116  10110  00 05 A0 FF 51 58 01  0558  1260
0116  DCR  E                  0117  10011  00 05 A0 FF 50 58 01  0558  1264
0117  DCR  H                  0118  10001  00 05 A0 FF 50 57 01  0558  1268
0118  DCR  L                  0119  11011  00 05 A0 FF 50 57 00  0558  1272
0119  DCR  M       5700 75  011A  10001  00 05 A0 FF 50 57 00  0558  1282

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
011A  DCR  M       5700 74  011B  10011  00 05 A0 FF 50 57 00  0558  1292
011B  MVI  A,22               011D  10011  22 05 A0 FF 50 57 00  0558  1299
011D  ADI  44                 011F  00010  66 05 A0 FF 50 57 00  0558  1306
011F  DAA                     0120  00010  66 05 A0 FF 50 57 00  0558  1310
0120  ADI  88                 0122  00110  EE 05 A0 FF 50 57 00  0558  1317
0122  DAA                     0123  10001  54 05 A0 FF 50 57 00  0558  1321
0123  CMA                     0124  10001  AB 05 A0 FF 50 57 00  0558  1325
0124  STC                     0125  10001  AB 05 A0 FF 50 57 00  0558  1329
0125  CMC                     0126  00001  AB 05 A0 FF 50 57 00  0558  1333
0126  CMC                     0127  10001  AB 05 A0 FF 50 57 00  0558  1337

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0127  LXI  B,1010             012A  10001  AB 10 10 FF 50 57 00  0558  1347
012A  LXI  D,5AA5             012D  10001  AB 10 10 5A A5 57 00  0558  1357
012D  DAD  B                  012E  00001  AB 10 10 5A A5 67 10  0558  1367
012E  DAD  D                  012F  00001  AB 10 10 5A A5 C1 B5  0558  1377
012F  DAD  H                  0130  10001  AB 10 10 5A A5 83 6A  0558  1387
0130  DAD  SP                 0131  00001  AB 10 10 5A A5 88 C2  0558  1397
0131  INX  B                  0132  00001  AB 10 11 5A A5 88 C2  0558  1403
0132  INX  D                  0133  00001  AB 10 11 5A A6 88 C2  0558  1409
0133  INX  H                  0134  00001  AB 10 11 5A A6 88 C3  0558  1415
0134  INX  SP                 0135  00001  AB 10 11 5A A6 88 C3  0559  1421

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0135  DCX  B                  0136  00001  AB 10 10 5A A6 88 C3  0559  1427
0136  DCX  D                  0137  00001  AB 10 10 5A A5 88 C3  0559  1433
0137  DCX  H                  0138  00001  AB 10 10 5A A5 88 C2  0559  1439
0138  DCX  SP                 0139  00001  AB 10 10 5A A5 88 C2  0558  1445
0139  MVI  A,5A               013B  00001  5A 10 10 5A A5 88 C2  0558  1452
013B  RLC                     013C  00001  B4 10 10 5A A5 88 C2  0558  1456
013C  RLC                     013D  10001  69 10 10 5A A5 88 C2  0558  1460
013D  RLC                     013E  00001  D2 10 10 5A A5 88 C2  0558  1464
013E  RLC                     013F  10001  A5 10 10 5A A5 88 C2  0558  1468
013F  RLC                     0140  10001  4B 10 10 5A A5 88 C2  0558  1472

 PC      INST       EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0140  RLC                     0141  00001  96 10 10 5A A5 88 C2  0558  1476
0141  RLC                     0142  10001  2D 10 10 5A A5 88 C2  0558  1480
0142  RLC                     0143  00001  5A 10 10 5A A5 88 C2  0558  1484
0143  RRC                     0144  00001  2D 10 10 5A A5 88 C2  0558  1488
0144  RRC                     0145  10001  96 10 10 5A A5 88 C2  0558  1492
0145  RRC                     0146  00001  4B 10 10 5A A5 88 C2  0558  1496
0146  RRC                     0147  10001  A5 10 10 5A A5 88 C2  0558  1500
0147  RRC                     0148  10001  D2 10 10 5A A5 88 C2  0558  1504
```

```
0149  RRC                     014A  10001  B4 10 10 5A A5 88 C2  0558  1512

 PC      INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
014A  RRC                     014B  00001  5A 10 10 5A A5 88 C2  0558  1516
014B  RAL                     014C  00001  B4 10 10 5A A5 88 C2  0558  1520
014C  RAL                     014D  10001  68 10 10 5A A5 88 C2  0558  1524
014D  RAL                     014E  00001  D1 10 10 5A A5 88 C2  0558  1528
014E  RAL                     014F  10001  A2 10 10 5A A5 88 C2  0558  1532
014F  RAL                     0150  10001  45 10 10 5A A5 88 C2  0558  1536
0150  RAL                     0151  00001  8B 10 10 5A A5 88 C2  0558  1540
0151  RAL                     0152  10001  16 10 10 5A A5 88 C2  0558  1544
0152  RAL                     0153  00001  2D 10 10 5A A5 88 C2  0558  1548
0153  RAL                     0154  00001  5A 10 10 5A A5 88 C2  0558  1552

 PC      INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0154  RAR                     0155  00001  2D 10 10 5A A5 88 C2  0558  1556
0155  RAR                     0156  10001  16 10 10 5A A5 88 C2  0558  1560
0156  RAR                     0157  00001  8B 10 10 5A A5 88 C2  0558  1564
0157  RAR                     0158  10001  45 10 10 5A A5 88 C2  0558  1568
0158  RAR                     0159  10001  A2 10 10 5A A5 88 C2  0558  1572
0159  RAR                     015A  00001  D1 10 10 5A A5 88 C2  0558  1576
015A  RAR                     015B  10001  68 10 10 5A A5 88 C2  0558  1580
015B  RAR                     015C  00001  B4 10 10 5A A5 88 C2  0558  1584
015C  RAR                     015D  00001  5A 10 10 5A A5 88 C2  0558  1588
015D  JMP  0162               0162  00001  5A 10 10 5A A5 88 C2  0558  1598

 PC      INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0162  NOP                     0163  00001  5A 10 10 5A A5 88 C2  0558  1602
0163  NOP                     0164  00001  5A 10 10 5A A5 88 C2  0558  1606
-* TEST JUMP INSTRUCTIONS
-SET  CY=0,P=0,Z=0,S=0
-TA  188H
0164  JC   0160               0167  00001  5A 10 10 5A A5 88 C2  0558  1613
0167  JZ   0160               016A  00001  5A 10 10 5A A5 88 C2  0558  1620
016A  JPE  0160               016D  00001  5A 10 10 5A A5 88 C2  0558  1627
016D  JM   0160               0170  00001  5A 10 10 5A A5 88 C2  0558  1634
0170  JNC  0176               0176  00001  5A 10 10 5A A5 88 C2  0558  1644
0176  JNZ  017C               017C  00001  5A 10 10 5A A5 88 C2  0558  1654
017C  JPO  0182               0182  00001  5A 10 10 5A A5 88 C2  0558  1664
0182  JP   0188               0188  00001  5A 10 10 5A A5 88 C2  0558  1674

 PC      INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0188  NOP                     0189  00001  5A 10 10 5A A5 88 C2  0558  1678
-SET  CY=1,P=1,Z=1,S=1
-TA  184H
0189  JNC  0160               018C  11111  5A 10 10 5A A5 88 C2  0558  1685
018C  JNZ  0160               018F  11111  5A 10 10 5A A5 88 C2  0558  1692
018F  JPO  0160               0192  11111  5A 10 10 5A A5 88 C2  0558  1699
0192  JP   0160               0195  11111  5A 10 10 5A A5 88 C2  0558  1706
0195  JC   019B               019B  11111  5A 10 10 5A A5 88 C2  0558  1716
019B  JZ   01A1               01A1  11111  5A 10 10 5A A5 88 C2  0558  1726
01A1  JPE  01A7               01A7  11111  5A 10 10 5A A5 88 C2  0558  1736
01A7  JM   01AD               01AD  11111  5A 10 10 5A A5 88 C2  0558  1746
01AD  LXI  H,01B4             01B0  11111  5A 10 10 5A A5 01 B4  0558  1756

 PC      INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
01B0  PCHL                    01B4  11111  5A 10 10 5A A5 01 B4  0558  1762
01B4  CALL 505B               505B  11111  5A 10 10 5A A5 01 B4  0556  1780
-* TEST CALL AND RETURN INSTRUCTIONS
-SET  CY=0,P=0,Z=0,S=0
-TA  1CFH
505B  MOV  A,B                505C  00001  10 10 10 5A A5 01 B4  0556  1784
505C  RET                     01B7  00001  10 10 10 5A A5 01 B4  0558  1794
01B7  CZ   505B               01BA  00001  10 10 10 5A A5 01 B4  0558  1803
01BA  CC   505B               01BD  00001  10 10 10 5A A5 01 B4  0558  1812
```

```
 01C0  CM    505D              01C3  00001  10 10 10 5A A5 01 B4  0558  1830
 01C3  CNZ   5058              5058  00001  10 10 10 5A A5 01 B4  0556  1848
 5058  MOV   A,B               505C  00001  10 10 10 5A A5 01 B4  0556  1852

  PC       INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
 505C  RET                     01C6  00001  10 10 10 5A A5 01 B4  0558  1862
 01C6  CNC   505B              5058  00001  10 10 10 5A A5 01 B4  0556  1880
 5058  MOV   A,B               505C  00001  10 10 10 5A A5 01 B4  0556  1884
 505C  RET                     01C9  00001  10 10 10 5A A5 01 B4  0558  1894
 01C9  CPO   505B              505B  00001  10 10 10 5A A5 01 B4  0556  1912
 5058  MOV   A,B               505C  00001  10 10 10 5A A5 01 B4  0556  1916
 505C  RET                     01CC  00001  10 10 10 5A A5 01 B4  0558  1926
 01CC  CP    505B              505B  00001  10 10 10 5A A5 01 B4  0556  1944
 505B  MOV   A,B               505C  00001  10 10 10 5A A5 01 B4  0556  1948
 505C  RET                     01CF  00001  10 10 10 5A A5 01 B4  0558  1958

  PC       INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
 01CF  NOP                     01D0  00001  10 10 10 5A A5 01 B4  0558  1962
 -SET  CY=1,P=1,Z=1,S=1
 -TA  1E8H
 01D0  CZ    505B              505B  11111  10 10 10 5A A5 01 B4  0556  1980
 505B  MOV   A,B               505C  11111  10 10 10 5A A5 01 B4  0556  1984
 505C  RET                     01D3  11111  10 10 10 5A A5 01 B4  0558  1994
 01D3  CC    505B              505B  11111  10 10 10 5A A5 01 B4  0556  2012
 505B  MOV   A,B               505C  11111  10 10 10 5A A5 01 B4  0556  2016
 505C  RET                     01D6  11111  10 10 10 5A A5 01 B4  0558  2026
 01D6  CPE   505B              505B  11111  10 10 10 5A A5 01 B4  0556  2044
 505B  MOV   A,B               505C  11111  10 10 10 5A A5 01 B4  0556  2048
 505C  RET                     01D9  11111  10 10 10 5A A5 01 B4  0558  2058

  PC       INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
 01D9  CM    505B              505B  11111  10 10 10 5A A5 01 B4  0556  2076
 505B  MOV   A,B               505C  11111  10 10 10 5A A5 01 B4  0556  2080
 505C  RET                     01DC  11111  10 10 10 5A A5 01 B4  0558  2090
 01DC  CNZ   505D              01DF  11111  10 10 10 5A A5 01 B4  0558  2099
 01DF  CNC   505D              01E2  11111  10 10 10 5A A5 01 B4  0558  2108
 01E2  CPO   505D              01E5  11111  10 10 10 5A A5 01 B4  0558  2117
 01E5  CP    505D              01E8  11111  10 10 10 5A A5 01 B4  0558  2126
 01E8  NOP                     01E9  11111  10 10 10 5A A5 01 B4  0558  2130
 -SET  CY=0,P=0,Z=0,S=0
 -TA  201H
 01E9  CALL 505E               505E  00001  10 10 10 5A A5 01 B4  0556  2148
 505E  RZ                      505F  00001  10 10 10 5A A5 01 B4  0556  2154

  PC       INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
 505F  RET                     01EC  00001  10 10 10 5A A5 01 B4  0558  2164
 01EC  CALL 5060               5060  00001  10 10 10 5A A5 01 B4  0556  2182
 5060  RC                      5061  00001  10 10 10 5A A5 01 B4  0556  2188
 5061  RET                     01EF  00001  10 10 10 5A A5 01 B4  0558  2198
 01EF  CALL 5062               5062  00001  10 10 10 5A A5 01 B4  0556  2216
 5062  RPE                     5063  00001  10 10 10 5A A5 01 B4  0556  2222
 5063  RET                     01F2  00001  10 10 10 5A A5 01 B4  0558  2232
 01F2  CALL 5064               5064  00001  10 10 10 5A A5 01 B4  0556  2250
 5064  RM                      5065  00001  10 10 10 5A A5 01 B4  0556  2256
 5065  RET                     01F5  00001  10 10 10 5A A5 01 B4  0558  2266

  PC       INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
 01F5  CALL 5066               5066  00001  10 10 10 5A A5 01 B4  0556  2284
 5066  RNZ                     01F8  00001  10 10 10 5A A5 01 B4  0558  2296
 01F8  CALL 5068               5068  00001  10 10 10 5A A5 01 B4  0556  2314
 5068  RNC                     01FB  00001  10 10 10 5A A5 01 B4  0558  2326
 01FB  CALL 506A               506A  00001  10 10 10 5A A5 01 B4  0556  2344
 506A  RPO                     01FE  00001  10 10 10 5A A5 01 B4  0558  2356
 01FE  CALL 506C               506C  00001  10 10 10 5A A5 01 B4  0556  2374
 506C  RP                      0201  00001  10 10 10 5A A5 01 B4  0558  2386
```

```
505E  RZ                            0204  11111  10 10 10 5A A5 01 B4  0558  2416

 PC       INST         EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0204  CALL 5060                5060  11111  10 10 10 5A A5 01 B4  0556  2434
5060  RC                       0207  11111  10 10 10 5A A5 01 B4  0558  2446
0207  CALL 5062                5062  11111  10 10 10 5A A5 01 B4  0556  2464
5062  RPE                      020A  11111  10 10 10 5A A5 01 B4  0558  2476
020A  CALL 5064                5064  11111  10 10 10 5A A5 01 B4  0556  2494
5064  RM                       020D  11111  10 10 10 5A A5 01 B4  0558  2506
020D  CALL 5066                5066  11111  10 10 10 5A A5 01 B4  0556  2524
5066  RNZ                      5067  11111  10 10 10 5A A5 01 B4  0556  2530
5067  RET                      0210  11111  10 10 10 5A A5 01 B4  0558  2540
0210  CALL 5068                5068  11111  10 10 10 5A A5 01 B4  0556  2558

 PC       INST         EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
5068  RNC                      5069  11111  10 10 10 5A A5 01 B4  0556  2564
5069  RET                      0213  11111  10 10 10 5A A5 01 B4  0558  2574
0213  CALL 506A                506A  11111  10 10 10 5A A5 01 B4  0556  2592
506A  RPO                      506B  11111  10 10 10 5A A5 01 B4  0556  2598
506B  RET                      0216  11111  10 10 10 5A A5 01 B4  0558  2608
0216  CALL 506C                506C  11111  10 10 10 5A A5 01 B4  0556  2626
506C  RP                       506D  11111  10 10 10 5A A5 01 B4  0556  2632
506D  RET                      0219  11111  10 10 10 5A A5 01 B4  0558  2642
0219  RST  00                  0000  11111  10 10 10 5A A5 01 B4  0556  2654
-T  2
0000  MVI  A,01                0002  11111  01 10 10 5A A5 01 B4  0556  2661

 PC       INST         EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0002  MVI  B,02                0004  11111  01 02 10 5A A5 01 B4  0556  2668
-DH
5065
01F5
5066
01F8
5068
01FB
506A
01FE
506C
0201
505E
0204
5060
0207
5062
020A
5064
020D
5066
5067
0210
5068
5069
0213
506A
506B
0216
506C
506D
0219
0000
0002
-* TEST NORMAL PORT I/O INSTRUCTIONS
```

```
*021A INPUT ON PORT 02 =
23H
021A  IN   02                 021C  11111  23 02 10 5A A5 01 B4  0556  2678
*021C INPUT ON PORT F0 =
45H
021C  IN   F0                 021E  11111  45 02 10 5A A5 01 B4  0556  2688
021E  MVI  A,28               0220  11111  28 02 10 5A A5 01 B4  0556  2695
*0220 OUTPUT ON PORT 05 = 28
0220  OUT  05                 0222  11111  28 02 10 5A A5 01 B4  0556  2705
0222  MVI  A,96               0224  11111  96 02 10 5A A5 01 B4  0556  2712
*0224 OUTPUT ON PORT E0 = 96
0224  OUT  E0                 0226  11111  96 02 10 5A A5 01 B4  0556  2722
-DI  2,0F0H
0002  23
00F0  45
-DO  5,0E0H
0005  28
00E0  96
-IP  2,0F0H
-OP  5,0E0H
-T  5
0226  IN   02                 0228  11111  23 02 10 5A A5 01 B4  0556  2732
0228  IN   F0                 022A  11111  45 02 10 5A A5 01 B4  0556  2742
022A  OUT  05                 022C  11111  45 02 10 5A A5 01 B4  0556  2752

 PC      INST         EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
022C  MVI  A,87               022E  11111  87 02 10 5A A5 01 B4  0556  2759
022E  OUT  E0                 0230  11111  87 02 10 5A A5 01 B4  0556  2769
-DI  2,0F0H
0002  23
00F0  45
-DO  5,0E0H
0005  45
00E0  87
-* TEST MEMORY MAPPED I/O INSTRUCTIONS
-MIC  5A00H
-MOC  5A00H
-T  5
*0230 INPUT ON MEMORY PORT 5A00 =
0A6H
0230  LDA  5A00      5A00 A6  0233  11111  A6 02 10 5A A5 01 B4  0556  2782
0233  LXI  H,5A00             0236  11111  A6 02 10 5A A5 5A 00  0556  2792
*0236 INPUT ON MEMORY PORT 5A00 =
0BFH
0236  MOV  A,M       5A00 BF  0237  11111  BF 02 10 5A A5 5A 00  0556  2799
*0237 OUTPUT ON MEMORY PORT 5A00 = BF
0237  STA  5A00      5A00 BF  023A  11111  BF 02 10 5A A5 5A 00  0556  2812
*023A OUTPUT ON MEMORY PORT 5A00 = A5
023A  MOV  M,E       5A00 A5  023B  11111  BF 02 10 5A A5 5A 00  0556  2819
-DM  5A00H
5A00  A5
-MIB  5A00H
-MOP  5A00H
-MDAT  5A00H 98H 0E1H
-SET  PC=230H
-T  5
0230  LDA  5A00      5A00 98  0233  11111  98 02 10 5A A5 5A 00  0556  2832
0233  LXI  H,5A00             0236  11111  98 02 10 5A A5 5A 00  0556  2842
0236  MOV  A,M       5A00 E1  0237  11111  E1 02 10 5A A5 5A 00  0556  2849

 PC      INST         EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0237  STA  5A00      5A00 E1  023A  11111  E1 02 10 5A A5 5A 00  0556  2862
023A  MOV  M,E       5A00 A5  023B  11111  E1 02 10 5A A5 5A 00  0556  2869
-DM  5A00H
```

```
-DIM
   IM = 00000000   SOD = 0  INT7.5 = 0
-T  3
023B  RIM                     023C  11111  00 02 10 5A A5 5A 00  0556  2873
023C  MVI  A,CF               023E  11111  CF 02 10 5A A5 5A 00  0556  2880
023E  SIM                     023F  11111  CF 02 10 5A A5 5A 00  0556  2884
-DIM
   IM = 00000111   SOD = 1  INT7.5 = 0
-SET  SI=1
-DIM
   IM = 10000111   SOD = 1  INT7.5 = 0
-T  1
023F  RIM                     0240  11111  87 02 10 5A A5 5A 00  0556  2888
-SIB
-SDAT  0 1 1 0
-T  5
0240  RIM                     0241  11111  07 02 10 5A A5 5A 00  0556  2892
0241  RIM                     0242  11111  87 02 10 5A A5 5A 00  0556  2896
0242  RIM                     0243  11111  87 02 10 5A A5 5A 00  0556  2900
0243  RIM                     0244  11111  07 02 10 5A A5 5A 00  0556  2904

 PC      INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
0244  RIM                     0245  11111  07 02 10 5A A5 5A 00  0556  2908
-SET  PC=240H
-T  5
0240  RIM                     0241  11111  87 02 10 5A A5 5A 00  0556  2912
0241  RIM                     0242  11111  87 02 10 5A A5 5A 00  0556  2916
0242  RIM                     0243  11111  07 02 10 5A A5 5A 00  0556  2920
0243  RIM                     0244  11111  07 02 10 5A A5 5A 00  0556  2924
0244  RIM                     0245  11111  87 02 10 5A A5 5A 00  0556  2928
-SIC
-SOC
-T  5
*0245 SERIAL INPUT =
1
0245  RIM                     0246  11111  87 02 10 5A A5 5A 00  0556  2932
0246  MVI  A,C0               0248  11111  C0 02 10 5A A5 5A 00  0556  2939
*0248 SERIAL OUTPUT = 1
0248  SIM                     0249  11111  C0 02 10 5A A5 5A 00  0556  2943
0249  MVI  A,80               024B  11111  80 02 10 5A A5 5A 00  0556  2950

 PC      INST        EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
024B  SIM                     024C  11111  80 02 10 5A A5 5A 00  0556  2954
-* TEST VARIOUS INTERRUPTS
-INT  I  7  0C3H 74H 01H
-SET  IE=1
-T  4
024C  MOV  A,B                024D  11111  02 02 10 5A A5 5A 00  0556  2958
 *** INTERRUPT RECOGNIZED
024D  MOV  D,B                024E  11111  02 02 10 02 A5 5A 00  0556  2962
024E  JMP  0174               0174  11111  02 02 10 02 A5 5A 00  0556  2972
0174  MOV  H,B                0175  11111  02 02 10 02 A5 02 00  0556  2976
-SET PC=24CH
-INT T 2
-T  4
 *** INTERRUPT RECOGNIZED
024C  MOV  A,B                0024  11111  02 02 10 02 A5 02 00  0554  2993
0024  MOV  B,D                0025  11111  02 02 10 02 A5 02 00  0554  2997
0025  MOV  C,D                0026  11111  02 02 02 02 A5 02 00  0554  3001
0026  MOV  D,D                0027  11111  02 02 02 02 A5 02 00  0554  3005
-SET PC=24CH
-* THIS INTERRUPT WILL NOT BE RECOGNIZED SINCE ENABLE BIT IS RESET
-INT  I 2 0
-T  4
```

```
 PC       INST          EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
024D  MOV  D,B                  024E  11111  02 02 02 02 A5 02 00  0554  3013
024E  NOP                       024F  11111  02 02 02 02 A5 02 00  0554  3017
024F  HLT                       0250  11111  02 02 02 02 A5 02 00  0554  3022
-SET PC=250H
-SET  IM=0
-DIM
    IM = 10000000   SOD = 1  INT7.5 = 0
-INT  7  0
-* THIS INTERRUPT WILL NOT BE RECOGNIZED UNTIL ENABLE BIT IS SET
-T  6
0250  MVI  C,51                 0252  11111  02 02 51 02 A5 02 00  0554  3029
0252  MVI  D,65                 0254  11111  02 02 51 65 A5 02 00  0554  3036
0254  EI                        0255  11111  02 02 51 65 A5 02 00  0554  3040
 *** INTERRUPT RECOGNIZED
0255  MOV  A,H                  003C  11111  02 02 51 65 A5 02 00  0552  3057
003C  MOV  E,L                  003D  11111  02 02 51 65 00 02 00  0552  3061
003D  MOV  H,L                  003E  11111  02 02 51 65 00 00 00  0552  3065
-DIM
    IM = 10000000   SOD = 1  INT7.5 = 1
-SET PC=25CH
-SET IE=1
-SET IM=1
-* THIS INTERRUPT WILL NOT BE RECOGNIZED BECAUSE IT IS MASKED
-INT  5  0
-T  1
025C  MOV  A,E                  025D  11111  00 02 51 65 00 00 00  0552  3069
-DIM
    IM = 10011001   SOD = 1  INT7.5 = 1
-* INTERRUPT 7.5 WILL NOT BE RECOGNIZED UNTIL INT7.5 FLIPFLOP IS RESET
-INT  7  0
-T  7

 PC       INST          EA (EA) NPC   CZSPI  A  B  C  D  E  H  L    SP   CYC
025D  LXI  B,0506               0260  11111  00 05 06 65 00 00 00  0552  3079
0260  RAL                       0261  01111  01 05 06 65 00 00 00  0552  3083
0261  DI                        0262  01111  01 05 06 65 00 00 00  0552  3087
0262  MOV  A,L                  0263  01111  00 05 06 65 00 00 00  0552  3091
0263  NOP                       0264  01111  00 05 06 65 00 00 00  0552  3095
0264  HLT                       0265  01111  00 05 06 65 00 00 00  0552  3100
-* SET SOME SYMBOL VALUES
-SSYM BEGIN=52, START=25H, STOP=100H
-DSYM
BEGIN   0034
START   0025
STOP    0100
-DSYM BEGIN
BEGIN   0034
-DM   START  BEGIN
0025  4A 52 5A 62   6A 7B 43 4B   53 5B 63
0030  6B 7C 44 4C   54
-DEL BEGIN
-DSYM
START   0025
STOP    0100
-SM  0FFF2H 25H
-DM  0FFF0H 0FFFFH
FFF0  76 76 25 76   76 76 76 76   76 76 76 76   76 76 76 76
-X
```

Figure 4-4

APPENDIX A

SIMULATOR MESSAGES

Simulator messages are divided into two classes; Command Mode messages and Execution Mode messages. Most messages indicate errors although some are merely informative. In the interactive mode, all error messages cause the Simulator to return to the Command Mode and cause the program to revert to using the standard I/O devices. In the batch mode, all errors cause the Simulation Program to terminate.

The following messages are considered as errors unless stated otherwise:

Command Messages

ADDRESS OUT OF RANGE - An operand that represents a memory address that is too large. The maximum memory address has been exceeded during a set memory command. The load address specified in an object module record is greater than available memory.

ARGUMENT ERROR - a command argument contains an invalid character. The user has specified a numeric that contains a character not valid for this numeric base.

CHECKSUM ERROR - object module contains a checksum error. User should reassemble source program to obtain new object module.

DATA TABLE ERROR - user has specified more data values than can be contained in the data table.

END OF FILE ENCOUNTERED - in the batch mode, an end-of-file (EOF) condition was detected while reading commands.

FILE NOT FOUND - a file name specified in the R, RD, RE, RED, W, L, or LS commands could not be found or opened.

INVALID CHARACTER - an invalid character was found while processing a command line.

INVALID COMMAND - the user specified command is not valid. See Command Summary.

INVALID ELEMENT - an invalid element was specified with the S or SET command.

INVALID OPERAND - a command operand was invalid.

LIMIT REACHED - This is an informative message only. It indicates that the number of instructions specified by the LI command has been executed. This message only occurs when the T, TA, TB, E, or EA command is used to initiate program execution.

MISSING OPERAND - the command requires an operand(s) but none was specified.

NUMBER OF BYTES READ = ____ - this is an informative message that indicates the number of bytes read in the object module by the L or LS commands. If the number of bytes read is zero, it probably indicates that an object module of the wrong format was read.

SYMBOL ERROR - a symbol in the object module was invalid. A symbol specified in the DSYM or SSYM commands started with a numeric character or contained an illegal character.

SYMBOL FORMAT ERROR - a symbol record in the object module specified a symbol with no corresponding value.

SYMBOL TABLE FULL - an attempt is made to place too many symbols into the symbol table. If this message occurs while reading an object module, it is an informative message only, but any remaining symbols are ignored. The user may increase the size of the symbol table.

SYNTAX ERROR - the user has specified an operand that contains invalid syntax. For example: 3+-5, LABEL/3

TERMINATOR ERROR - an invalid terminator was specified for an operand. E.G.   SSYM   LABEL*- 56H

UNDEFINED SYMBOL - a symbolic operand was specified that is not in the symbol table.

VALUE OUT OF RANGE - a value has been specified that is too large. A byte value is greater than 255. An element consisting of 1 bit is greater than 1.

Execution Messages

ADDRESS OUT OF RANGE - the program counter exceeds the legal maximum.

DATA TABLE ERROR - an attempt has been made to read data from the data buffer table for a port for which no data has been defined by the DATA command. In the batch mode, this is an informative message only; the value of the accumulator does not change and execution proceeds. In the interactive mode, this is an error message.

ILLEGAL INSTRUCTION - an attempt has been made to execute an illegal instruction.

INVALID INPUT DATA - the user has entered input data for an input port that is out of range or is an illegal numeric. This message will only appear in the batch mode or in a read mode.

WRITING TO PROTECTED MEMORY - the Simulator has executed a microprocessor instruction that writes to protected memory.

APPENDIX B

COMMAND SUMMARY

The following list is a summary of the 8080/8085 Simulator commands.

*	- Comment
;	- Comment
BASE	- Set Numeric Input Base
BP	- Set Address Breakpoint
DATA	- Specify Input Buffer Data for Normal I/O Ports
DC	- Display CPU Status
DEL	- Delete Symbols
DH	- Display History
DIM	- Display Interrupt Mask
DIN	- Display Input Port
DM	- Display Program Memory
DOUT	- Display Output Port
DSYM	- Display Symbols
E	- Execute Instructions
EA	- Execute Instructions Until Address
FIN	- Fill Input Port
FM	- Fill Program Memory
FORM	- Set Display Line Format
FOUT	- Fill Output Port
H	- Specify Heading Count
IB	- Read Port Input Data from Data Buffer
IC	- Read Port Input Data from Current Input Device
INT	- Set Instruction Interrupt
IP	- Read Port Input Data from Preset Data Latch
IS	- Read Port Input Data from Standard Input Device
L	- Load Object Module
LI	- Set Instruction Execution Limit
LS	- Load Object Module with Symbol Table

MDAT	Specify Input Buffer Data for Memory Mapped I/O ports
MIB	Read Memory Input Port Data from Data Buffer
MIC	Read Memory Input Port Data from Current Input Device
MIP	Read Memory Input Port Data from Preset Data Latch
MIS	Read Memory Input Port Data from Standard Input Device
MOC	Write Memory Mapped Output Port Data to Current Output Device
MOP	Write Memory Mapped Output Port Data to Data Latch
MOS	Write Memory Mapped Output Port Data to Standard Output Device
NBP	Clear Address Breakpoint
NINT	Clear Instruction Interrupt
NPRO	Clear Memory Protect Flags
NTR	Clear Address Trace Flags
OC	Write Output Data to Current Output Device
OP	Write Output Data to Data Latch
OS	Write Output Data to Standard Output Device
PRO	Set Protect Flag for Address Range
R	Read Commands from Alternate Input Device or File
RD	Read Commands with Delay
RE	Read Commands with Echo
RED	Read Commands with Echo and Delay
RES	Reset Microprocessor
RET	Return from Read Mode, Read Commands from Standard Input Device
SDAT	Specify Input Buffer Data for Serial Port
SET	Set Processor Element (Registers, Status Bits, etc.)
SIB	Read Serial Input Port Data from Data Buffer
SIC	Read Serial Input Port Data from Current Input Device
SIN	Set Input Port Data Value
SIP	Read Serial Input Port Data from Preset Data Latch
SIS	Read Serial Input Port Data from Standard Input Device
SM	Set Program Memory

SOC	- Write Serial Output Port Data to Current Output Device
SOP	- Write Serial Output Port Data to Data Latch
SOS	- Write Serial Output Port Data to Standard Output Device
SOUT	- Set Output Port Data Value
SSYM	- Set Symbols
T	- Trace Instructions
TA	- Trace Instructions until Address
TB	- Trace Instructions with Breakpoints
TR	- Trace Flags for Address Range
TYPE	- Specify Processor Type
W	- Write Output to Alternate Device or File
X	- Exit Simulator

APPENDIX C

OBJECT MODULE FORMATS

The object module is a machine readable computer output in the form of punched cards, paper tape, etc. The object module contains specifications for loading the memory of the target microprocessor. The object module is produced as a series of card images by Microtec's 8080/8085 Macro Assembler or any other compatible assembler. Each object record contains the load address and data specifications for up to 255 bytes of data. Symbol table information may also be included. The format of an object module is shown below.

```
$$
 symbol records
$$
 data records
```

A record consisting of two dollar signs indicates symbol records follow. A sample symbol record is shown below:

```
APPLE   ØØØØØH   LABEL1   ØDØC3H   MEM      ØFFFFH
```

A symbol record consists of up to four symbols, with each symbol's value immediately following the symbol. The symbol and symbol values must be separated by at least one blank. If the symbol's value is in a base other than decimal, a single letter descriptor must follow the value; "H" for hexadecimal, "Q" for Octal. A second record consisting of two dollar signs follows the last symbol record.

The format of a data record is shown below.

```
1  2  3  4  5  6  7  8  9  10 11  ...  40 41 42 43
:  byte     load     type  data        data checksum
   count    address
```

Column 1 contains the code for a colon. This marks the beginning of an object data record.

Column 2 and 3 contain the count of the number of data bytes contained in the record. If this field contains an "ØØ" it signifies the end of the object module.

Columns 4 through 7 contain the load address expressed as hexadecimal digits. The first data byte is to be loaded into this address, subsequent data bytes into the next sequential addresses. Columns 4 and 5 contain the most significant byte of the address.

Columns 8 and 9 contain the record type. Presently two types are defined. "ØØ" indicates a data record. "Ø1" indicates a terminator record. In this case the byte count will also be zero and the load address will contain the module starting address.

Columns 10 to 41 (more or less depending upon number of data bytes) contain the hexadecimal specifications for up to 16 bytes of data.

The last two columns in the record contain a checksum. The checksum is the negative of the sum of all bytes in the record (except column 1) evaluated modulo 256. Thus when the record is read, the sum of all bytes, including the checksum, should be zero.

APPENDIX D

8080/8085 OPERATION CODES

The following table illustrates the proper format for writing 8080/8085 instructions. The operation code mnemonics listed are the only valid opcodes for the assembler.

These symbols are used in the table.

D,S - indicates a source or destination register which is one of the following: A,B,C,D,E,H,L,M

RP - indicates a register pair which may be one of the following: B,D,H,SP

PSW - indicates the Program Status Word

exp_8 - indicates an 8 bit value

exp_{16} - indicates a 16 bit value

ddd
sss - the bit pattern representing one of the registers denoted by D or S above. The bit patterns are as follows:

B - 000 C - 001 D - 010
E - 011 H - 100 L - 101
M - 110 A - 111

rp - the bit pattern representing one of the register pairs denoted by RP above. The bit patterns are as follows:

B - 00 D - 01 H - 10 SP - 11

* - new instruction of 8085

When two states are shown for an instruction, the first number is if the condition is not satisfied and the second number is if the condition is satisfied.

SYMBOLIC OPCODE	FIRST BYTE MACHINE CODE	NUMBER OF BYTES	NUMBER OF STATES	
			8080	8085
Data Transfer				
MOV D,S	01dddsss	1	5	4
MOV D,M	01ddd110	1	7	7
MOV M,S	01110sss	1	7	7
MVI D,exp_8	00ddd110	2	7	7
MVI M,exp_8	00110110	2	10	10
LXI RP,exp_{16}	00rp0001	3	10	10
LDA exp_{16}	00111010	3	13	13
STA exp_{16}	00110010	3	13	13
LHLD exp_{16}	00101010	3	16	16
SHLD exp_{16}	00100010	3	16	16
LDAX RP	00rp1010	1	7	7
STAX RP	00rp0010	1	7	7
XCHG	11101011	1	4	4
Arithmetic Group				
ADD S	10000sss	1	4	4
ADC S	10001sss	1	4	4
SUB S	10010sss	1	4	4
SBB S	10011sss	1	4	4
ADI exp_8	11000110	2	7	7
ACI exp_8	11001110	2	7	7
SUI exp_8	11010110	2	7	7
SBI exp_8	11011110	2	7	7
INR D	00ddd100	1	5	4
DCR D	00ddd101	1	5	4
INX RP	00rp0011	1	5	6
DCX RP	00rp1011	1	5	6
DAD RP	00rp1001	1	10	10
DAA	00100111	1	4	4
Logical Group				
ANA S	10100sss	1	4	4
XRA S	10101sss	1	4	4
ORA S	10110sss	1	4	4
CMP S	10111sss	1	4	4
ANI exp_8	11100110	2	7	7
XRI exp_8	11101110	2	7	7
ORI exp_8	11110110	2	7	7
CPI exp_8	11111110	2	7	7
RLC	00000111	1	4	4
RRC	00001111	1	4	4
RAL	00010111	1	4	4
RAR	00011111	1	4	4
CMA	00101111	1	4	4
CMC	00111111	1	4	4
STC	00110111	1	4	4

SYMBOLIC OPCODE	FIRST BYTE MACHINE CODE	NUMBER OF BYTES	NUMBER OF STATES	
			8080	8085
Branch Group				
JMP exp_{16}	11000011	3	10	10
JNZ exp_{16}	11000010	3	10	7/10
JZ exp_{16}	11001010	3	10	7/10
JNC exp_{16}	11010010	3	10	7/10
JC exp_{16}	11011010	3	10	7/10
JPO exp_{16}	11100010	3	10	7/10
JPE exp_{16}	11101010	3	10	7/10
JP exp_{16}	11110010	3	10	7/10
JM exp_{16}	11111010	3	10	7/10
CALL exp_{16}	11001101	3	17	18
CNZ exp_{16}	11000100	3	11/17	9/18
CZ exp_{16}	11001100	3	11/17	9/18
CNC exp_{16}	11010100	3	11/17	9/18
CC exp_{16}	11011100	3	11/17	9/18
CPO exp_{16}	11100100	3	11/17	9/18
CPE exp_{16}	11101100	3	11/17	9/18
CP exp_{16}	11110100	3	11/17	9/18
CM exp_{16}	11111100	3	11/17	9/18
RET	11001001	1	10	10
RNZ	11000000	1	5/11	6/12
RZ	11001000	1	5/11	6/12
RNC	11010000	1	5/11	6/12
RC	11011000	1	5/11	6/12
RPO	11100000	1	5/11	6/12
RPE	11101000	1	5/11	6/12
RP	11110000	1	5/11	6/12
RM	11111000	1	5/11	6/12
RST A	11aaa111	1	11	12
PCHL	11101001	1	5	6
Stack, I/O and Machine Control Group				
PUSH RP	11rp0101	1	11	12
PUSH PSW	11110101	1	11	12
POP RP	11rp0001	1	10	10
POP PSW	11110001	1	10	10
XTHL	11100011	1	18	16
SPHL	11111001	1	5	6
IN exp_8	11011011	2	10	10
OUT exp_8	11010011	2	10	10
EI	11111011	1	4	4
DI	11110011	1	4	4
HLT	01110110	1	7	5
NOP	00000000	1	4	4
RIM	00100000	1	*	4
SIM	00110000	1	*	4

Appendix E

HEXADECIMAL-DECIMAL CONVERSION TABLE

This table allows conversions to be made between hexadecimal and decimal numbers. The table has a decimal range of 0 to 4095. To convert larger numbers add the following values to the table values.

Hexadecimal	Decimal
1000	4096
2000	8192
3000	12228
4000	16384
5000	20480
6000	24576
7000	28672
8000	32768
9000	36864
A000	40960
B000	45056
C000	49152
D000	53248
E000	57344
F000	61440

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000	0000	0001	0002	0003	0004	0005	0006	0007	0008	0009	0010	0011	0012	0013	0014	0015
010	0016	0017	0018	0019	0020	0021	0022	0023	0024	0025	0026	0027	0028	0029	0030	0031
020	0032	0033	0034	0035	0036	0037	0038	0039	0040	0041	0042	0043	0044	0045	0046	0047
030	0048	0049	0050	0051	0052	0053	0054	0055	0056	0057	0058	0059	0060	0061	0062	0063
040	0064	0065	0066	0067	0068	0069	0070	0071	0072	0073	0074	0075	0076	0077	0078	0079
050	0080	0081	0082	0083	0084	0085	0086	0087	0088	0089	0090	0091	0092	0093	0094	0095
060	0096	0097	0098	0099	0100	0101	0102	0103	0104	0105	0106	0107	0108	0109	0110	0111
070	0112	0113	0114	0115	0116	0117	0118	0119	0120	0121	0122	0123	0124	0125	0126	0127
080	0128	0129	0130	0131	0132	0133	0134	0135	0136	0137	0138	0139	0140	0141	0142	0143
090	0144	0145	0146	0147	0148	0149	0150	0151	0152	0153	0154	0155	0156	0157	0158	0159
0A0	0160	0161	0162	0163	0164	0165	0166	0167	0168	0169	0170	0171	0172	0173	0174	0175
0B0	0176	0177	0178	0179	0180	0181	0182	0183	0184	0185	0186	0187	0188	0189	0190	0191
0C0	0192	0193	0194	0195	0196	0197	0198	0199	0200	0201	0202	0203	0204	0205	0206	0207
0D0	0208	0209	0210	0211	0212	0213	0214	0215	0216	0217	0218	0219	0220	0221	0222	0223
0E0	0224	0225	0226	0227	0228	0229	0230	0231	0232	0233	0234	0235	0236	0237	0238	0239
0F0	0240	0241	0242	0243	0244	0245	0246	0247	0248	0249	0250	0251	0252	0253	0254	0255

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
100	0256	0257	0258	0259	0260	0261	0262	0263	0264	0265	0266	0267	0268	0269	0270	0271
110	0272	0273	0274	0275	0276	0277	0278	0279	0280	0281	0282	0283	0284	0285	0286	0287
120	0288	0289	0290	0291	0292	0293	0294	0295	0296	0297	0298	0299	0300	0301	0302	0303
130	0304	0305	0306	0307	0308	0309	0310	0311	0312	0313	0314	0315	0316	0317	0318	0319
140	0320	0321	0322	0323	0324	0325	0326	0327	0328	0329	0330	0331	0331	0333	0334	0335
150	0336	0337	0338	0339	0340	0341	0342	0343	0344	0345	0346	0347	0348	0349	0350	0351
160	0352	0353	0354	0355	0356	0357	0358	0359	0360	0361	0362	0363	0364	0365	0366	0367
170	0368	0369	0370	0371	0372	0373	0374	0375	0376	0377	0378	0379	0380	0381	0382	0383
180	0384	0385	0386	0387	0388	0389	0390	0391	0392	0393	0394	0395	0396	0397	0398	0399
190	0400	0401	0402	0403	0404	0405	0406	0407	0408	0409	0410	0411	0412	0413	0414	0415
1A0	0416	0417	0418	0419	0420	0421	0422	0423	0424	0425	0426	0427	0428	0429	0430	0431
1B0	0432	0433	0434	0435	0436	0437	0438	0439	0440	0441	0442	0443	0444	0445	0446	0447
1C0	0448	0449	0450	0451	0452	0453	0454	0455	0456	0457	0458	0459	0460	0461	0462	0463
1D0	0464	0465	0466	0467	0468	0469	0470	0471	0472	0473	0474	0475	0476	0477	0478	0479
1E0	0480	0481	0482	0483	0484	0485	0486	0487	0488	0489	0490	0491	0492	0493	0494	0495
1F0	0496	0497	0498	0499	0500	0501	0502	0503	0504	0505	0506	0507	0508	0509	0510	0511
200	0512	0513	0514	0515	0516	0517	0518	0519	0520	0521	0522	0523	0524	0525	0526	0527
210	0528	0529	0530	0531	0532	0533	0534	0535	0536	0537	0538	0539	0540	0541	0542	0543
220	0544	0545	0546	0547	0548	0549	0550	0551	0552	0553	0554	0555	0556	0557	0558	0559
230	0560	0561	0562	0563	0564	0565	0566	0567	0568	0569	0570	0571	0572	0573	0574	0575
240	0576	0577	0578	0579	0580	0581	0582	0583	0584	0585	0586	0587	0588	0589	0590	0591
250	0592	0593	0594	0595	0596	0597	0598	0599	0600	0601	0602	0603	0604	0605	0606	0607
260	0608	0609	0610	0611	0612	0613	0614	0615	0616	0617	0618	0619	0620	0621	0622	0623
270	0624	0625	0626	0627	0628	0629	0630	0631	0632	0633	0634	0635	0636	0637	0638	0639
280	0640	0641	0642	0643	0644	0645	0646	0647	0648	0649	0650	0651	0652	0653	0654	0655
290	0656	0657	0658	0659	0660	0661	0662	0663	0664	0665	0666	0667	0668	0669	0670	0671
2A0	0672	0673	0674	0675	0676	0677	0678	0679	0680	0681	0682	0683	0684	0685	0686	0687
2B0	0688	0689	0690	0691	0692	0693	0694	0695	0696	0697	0698	0699	0700	0701	0702	0703
2C0	0704	0705	0706	0707	0708	0709	0710	0711	0712	0713	0714	0715	0716	0717	0718	0719
2D0	0720	0721	0722	0723	0724	0725	0726	0727	0728	0729	0730	0731	0732	0733	0734	0735
2E0	0736	0737	0738	0739	0740	0741	0742	0743	0744	0745	0746	0747	0748	0749	0750	0751
2F0	0752	0753	0754	0755	0756	0757	0758	0759	0760	0761	0762	0763	0764	0765	0766	0767
300	0768	0769	0770	0771	0772	0773	0774	0775	0776	0777	0778	0779	0780	0781	0782	0783
310	0784	0785	0786	0787	0788	0789	0790	0791	0792	0793	0794	0795	0796	0797	0798	0799
320	0800	0301	0802	0803	0804	0805	0806	0807	0808	0809	0810	0811	0812	0813	0814	0815
330	0816	0817	0818	0819	0820	0821	0822	0823	0824	0825	0826	0827	0828	0829	0830	0831
340	0832	0833	0834	0835	0836	0837	0838	0839	0840	0841	0842	0843	0844	0845	0846	0847
350	0848	0849	0850	0851	0852	0853	0854	0855	0856	0857	0858	0859	0860	0861	0862	0863
360	0864	0865	0866	0867	0868	0869	0870	0871	0872	0873	0874	0875	0876	0877	0878	0879
370	0880	0881	0882	0883	0884	0885	0886	0887	0888	0889	0890	0891	0892	0893	0894	0895
380	0896	0897	0898	0899	0900	0901	0902	0903	0904	0905	0906	0907	0908	0909	0910	0911
390	0212	0913	0914	0915	0916	0917	0918	0919	0920	0921	0922	0923	0924	0925	0926	0927
3A0	0928	0929	0930	0931	0932	0933	0934	0935	0936	0937	0938	0939	0940	0941	0942	0943
3B0	0944	0945	0946	0947	0948	0949	0950	0951	0952	0953	0954	0955	0956	0957	0958	0959
3C0	0960	0961	0962	0963	0964	0965	0966	0967	0968	0969	0970	0971	0972	0973	0974	0975
3D0	0976	0977	0978	0979	0980	0981	0982	0983	0984	0985	0986	0987	0988	0989	0990	0991
3E0	0992	0993	0994	0995	0996	0997	0998	0999	1000	1001	1002	1003	1004	1005	1006	1007
3F0	1008	1009	1010	1011	1012	1013	1014	1015	1016	1017	1018	1019	1020	1021	1022	1023

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
400	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039
410	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055
420	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071
430	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087
440	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
450	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119
460	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
470	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151
480	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167
490	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
4A0	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199
4B0	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215
4C0	1216	1217	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1229	1230	1231
4D0	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247
4E0	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
4F0	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279
500	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295
510	1296	1297	1298	1299	1300	1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311
520	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327
530	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
540	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359
550	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375
560	1376	1377	1378	1379	1380	1381	1382	1383	1384	1385	1386	1387	1388	1389	1390	1391
570	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407
580	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
590	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439
5A0	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455
5B0	1456	1457	1458	1459	1460	1461	1462	1463	1464	1465	1466	1467	1468	1469	1470	1471
5C0	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487
5D0	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
5E0	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519
5F0	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535
600	1536	1537	1538	1539	1540	1541	1542	1543	1544	1545	1546	1547	1548	1549	1550	1551
610	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567
620	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
630	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599
640	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615
650	1616	1617	1618	1619	1620	1621	1622	1623	1624	1625	1626	1627	1628	1629	1630	1631
660	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647
670	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
680	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679
690	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695
6A0	1696	1697	1698	1699	1700	1701	1702	1703	1704	1705	1706	1707	1708	1709	1710	1711
6B0	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727
6C0	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
6D0	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759
6E0	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775
6F0	1776	1777	1778	1779	1780	1781	1782	1783	1784	1785	1786	1787	1788	1789	1790	1791

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807
10	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
20	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839
30	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855
40	1856	1857	1858	1859	1860	1861	1862	1863	1864	1865	1866	1867	1868	1869	1870	1871
50	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887
60	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
70	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919
80	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935
90	1936	1937	1938	1939	1940	1941	1942	1943	1944	1945	1946	1947	1948	1949	1950	1951
A0	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967
B0	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
C0	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999
D0	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
E0	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031
F0	2032	2033	2034	2035	2036	2037	2038	2039	2040	2041	2042	2043	2044	2045	2046	2047
800	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063
810	2064	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079
820	2080	2081	2082	2083	2084	2085	2086	2087	2088	2089	2090	2091	2092	2093	2094	2095
830	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111
840	2112	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127
850	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143
860	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159
870	2160	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175
880	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189	2190	2191
890	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207
8A0	2208	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223
8B0	2224	2225	2226	2227	2228	2229	2230	2231	2232	2233	2234	2235	2236	2237	2238	2239
8C0	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255
8D0	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271
8E0	2272	2273	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287
8F0	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303
900	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319
910	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335
920	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351
930	2352	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367
940	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378	2379	2380	2381	2382	2383
950	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
960	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415
970	2416	2417	2418	2419	2420	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431
980	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447
990	2448	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463
9A0	2464	2465	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475	2476	2477	2478	2479
9B0	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495
9C0	2496	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511
9D0	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527
9E0	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543
9F0	2544	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A00	2560	2561	2562	2563	2564	2565	2566	2567	2568	2569	2570	2571	2572	2573	2574	2575
A10	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591
A20	2592	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607
A30	2608	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2619	2620	2621	2622	2623
A40	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639
A50	2640	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655
A60	2656	2657	2658	2659	2660	2661	2662	2663	2664	2665	2666	2667	2668	2669	2670	2671
A70	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687
A80	2688	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703
A90	2704	2705	2706	2707	2708	2709	2710	2711	2712	2713	2714	2715	2716	2717	2718	2719
AA0	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735
AB0	2736	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751
AC0	2752	2753	2754	2755	2756	2757	2758	2759	2760	4761	2762	2763	2764	2765	2766	2767
AD0	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783
AE0	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799
AF0	2800	2801	2802	2803	2804	2805	2806	2807	2808	2809	2810	2811	2812	2813	2814	2815
B00	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831
B10	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
B20	2848	2849	2850	3851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863
B30	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879
B40	2880	2881	2882	2883	2884	2885	2866	2887	2888	2889	2890	2891	2892	2893	2894	2895
B50	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911
B60	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
B70	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943
B80	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959
B90	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975
BA0	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991
B80	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
BC0	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023
BD0	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039
BE0	3040	3041	3042	3043	3044	3045	3046	3047	3048	3049	3050	3051	3052	3053	3054	3055
BF0	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071
C00	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
C10	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103
C20	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119
C30	3120	3121	3122	3123	3124	3125	3126	3127	3128	3129	3130	3131	3132	3133	3134	3135
C40	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151
C50	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
C60	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183
C70	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199
C80	3200	3201	3202	3203	3204	3205	3206	3207	3208	3209	3210	3211	3212	3213	3214	3215
C90	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231
CA0	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
CB0	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263
CC0	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279
CD0	3280	3281	3282	3283	3284	3285	3286	3287	3288	3289	3290	3291	3292	3293	3294	3295
CE0	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311
CF0	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327

HEXADECIMAL-DECIMAL INTEGER CONVERSION (Cont'd)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
D00	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343
D10	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359
D20	3360	3361	3362	3363	3364	3365	3366	3367	3368	3369	3370	3371	3372	3373	3374	3375
D30	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391
D40	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
D50	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423
D60	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439
D70	3440	3441	3442	3443	3444	3445	3446	3447	3448	3449	3450	3451	3452	3453	3454	3455
D80	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471
D90	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
DA0	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503
DB0	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519
DC0	3520	3521	3522	3523	3524	3525	3526	3527	3528	3529	3530	3531	3532	3533	3534	3535
CC0	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551
DE0	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
DF0	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583
E00	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599
E10	3600	3601	3602	3603	3604	3605	3606	3607	3608	3609	3610	3611	3612	3613	3614	3615
E20	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631
E30	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
E40	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
E50	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679
E60	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693	3694	3695
E70	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711
E80	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
E90	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743
EA0	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759
EB0	3760	3761	3762	3763	3764	3765	3766	3767	3768	3769	3770	3771	3772	3773	3774	3775
EC0	3776	3777	3778	3779	3780	3781	3782	3783	3784	3785	3786	3787	3788	3789	3790	3791
ED0	3792	3793	3794	3795	3796	3797	3798	3799	3800	3801	3802	3803	3804	3805	3806	3807
EE0	3808	3809	3810	3811	3812	3813	3814	3815	3816	3817	3818	3819	3820	3821	3822	3823
EF0	3824	3825	3826	3827	3828	3829	3830	3831	3832	3833	3834	3835	3836	3837	3838	3839
F00	3840	3841	3842	3843	3844	3845	3846	3847	3848	3849	3850	3851	3852	3853	3854	3855
F10	3856	3857	3858	3859	3860	3861	3862	3863	3864	3865	3866	3867	3868	3869	3870	3871
F20	3872	3873	3874	3875	3876	3877	3878	3879	3880	3881	3882	3883	3884	3885	3886	3887
F30	3888	3889	3890	3891	3892	3893	3894	3895	3896	3897	3898	3899	3900	3901	3902	3903
F40	3904	3905	3906	3907	3908	3909	3910	3911	3912	3913	3914	3915	3916	3917	3918	3919
F50	3920	3921	3922	3923	3924	3925	3926	3927	3928	3929	3930	3931	3932	3933	3934	3935
F60	3936	3937	3938	3939	3940	3941	3942	3943	3944	3945	3946	3947	3948	3949	3950	3951
F70	3952	3953	3954	3955	3956	3957	3958	3959	3960	3961	3962	3963	3964	3965	3966	3967
F80	3968	3969	3970	3971	3972	3973	3974	3975	3976	3977	3978	3979	3980	3981	3982	3983
F90	3984	3985	3986	3987	3988	3989	3990	3991	3992	3993	3994	3995	3996	3997	3998	3999
FA0	4000	4001	4002	4003	4004	4005	4006	4007	4008	4009	4010	4011	4012	4013	4014	4015
FB0	4016	4017	4018	4019	4020	4021	4022	4023	4024	4025	4026	4027	4028	4029	4030	4031
FC0	4032	4033	4034	4035	4036	4037	4038	4039	4040	4041	4042	4043	4044	4045	4046	4047
FD0	4048	4049	4050	4051	4052	4053	4054	4055	4056	4057	4058	4059	4060	4061	4062	4063
FE0	4064	4065	4066	4067	4068	4069	4070	4071	4072	4073	4074	4075	4076	4077	4078	4079
FF0	4080	4081	4082	4083	4084	4085	4086	4087	4088	4089	4090	4091	4092	4093	4094	4095

SIMULATOR INSTALLATION NOTES

These notes are designed to help the user install the Simulator and perform any modifications that may be necessary for a particular computer. The Notes are separated into six sections: Program Installation, Program Modifications, Batch/Interactive Mode, Program Input/Output, Memory Requirements and Overlays, and NOVA Modifications.

A. Program Installation

1. The Simulator should be compiled once and its object module stored on some secondary storage device (disk). Compile the program in the usual manner, assigning it a name which can be refered to by an Execute or Run Statement. It is usually helpful to compile each subroutine separately. If upon loading the compiled program, it is discovered that not enough main memory is available to hold the entire program, refer to the section describing overlay structures.

B. Program Modifications

1. Some computers do not accept the full ASCII character set. Therefore, some of the characters defined in Subroutine INIT may be illegal and give a compilation error. If this is the case on your computer, the illegal characters must be replaced by legal characters. The characters are in the Array NALPH. If the illegal character is not used in the simulator as an operator, terminator, or a character in a symbol, replace the illegal character with a zero, 0. The illegal character may not be used between quote marks to represent an ASCII character constant. If the illegal character is used by the Simulator, replace the character with a unique legal character and use the new character in place of the old, illegal character. Note: some computers will not accept certain characters during a Fortran compilation, but will accept the characters as program input data.

In this case, the user could define the problem characters as numbers instead of hollerith constants. The numbers used would be the internal values of the characters as they would appear in a 1H data specification. An example of characters defined in this manner is shown in the NOVA modifications.

2. The variable IBIT corresponds to the number of bits per word in the host computer. IBIT is initially set to 16. This variable determines how many characters are packed into one host computer word for symbols stored in the Simulator symbol table. The user may want to increase this variable if the computer has a longer word length. However, it is not necessary. Increasing IBIT will allow a larger number of symbols to be stored in a fixed amount of memory. When initially installing the program, it is suggested that IBIT be left at 16 until the program is known to be operating correctly.

3. To increase the size of the symbol table and thus the number and length of the symbols the symbol table can hold, the user must change certain variables. The variables that must be changed depend on the number of bits per host computer word (see 2), the number of symbols in the symbol table, and the number of characters used to define a symbol. The variables that define these parameters are described below.

IBIT - number of bits per host computer word (set by user)
MLAB - maximum label length in characters (set by user)
ICCNT - number of characters per host computer word (calculated)
IWORD - number of computer words per symbol (calculated)
LTAB - length of symbol table (set by user)

The user must change the following variables to reflect the size of the symbol table and length of a symbol. The length of a symbol should probably correspond to the length set in the Assembler if symbols are passed from the assembler. The arrays are in COMMON, and therefore, the dimensions need to be changed in every subroutine and the main program.

```
    ITAB(IWORD,LTAB)                  where: IWORD = 1+(MLAB-1)/ICCNT
    ITAV(LTAB)                               ICCNT = IBIT/8
     NAME(IWORD)
```

4. The Simulator uses a random access disk file to simulate the full 65536 bytes of microprocessor memory. The memory sections or pages most recently accessed by the simulated program are swapped into a main memory array. This procedure minimizes the memory requirements of the simulator on the host computer. A multi-page scheme keeps page swapping to a minimum. Several other things have been done to minimize page swapping and keep program execution speed high. Memory pages are initialized only when they are accessed. If they are never accessed, they are never initialized. If the data on a page is not changed, the page is not rewritten to the disk file since this is not necessary and would only slow the program down.

If the user wishes to, he may increase or decrease the memory page size and the record length of the simulated memory disk file. If desired, the whole simulated memory may be implemented in main memory, eliminating the intermediate disk file altogether. If the user does perform any of these modifications, he must be aware of the following key variables.

MXMEM - maximum memory size simulated
(initially set to 65536, set by user)

MEM - array used to hold simulated pages in main memory
(initially set to 1536, set by user to 3*MSIZE)

MSIZE - length of memory page
(initially set to 512, set by user)

IRLEN - length of simulated memory disk file record in words
(initially set to 128, set by user)

INUM - number of disk records per memory page
(initially 4, calculated variable, INUM = MSIZE/IRLEN)

KPAGE - array indicating whether a page has been accessed
(initially set to 128, set by user to 65536/MSIZE)

Because the disk physical record size for some computers is limited, each disk read or write transfers only 128 words of simulated memory. Therefore, when a 512 byte page is swapped, 4 disk transfers take place. If larger records can be handled on the user's computer, disk activity can be minimized (and execution time reduced) by increasing the number of words per disk read and write. The record length should be a power of two and evenly divisible into the page size. If possible, increase the IRLEN variable to the page size, MSIZE.

The user should carefully consider the mater before increasing or decreasing the page size of the simulated memory. Increasing the page size may speed the simulator up, it could also slow it down. Likewise, decreasing the page size may also affect the speed in either direction. What happens to execution time when the page size changes depends on the program being simulated. If the user only simulates programs of 1K words in length and the data page is 256 bytes, then a page size of 512 words is perfect. Larger pages are not needed, and smaller pages would only increase disk activity. If the user simulates programs that jump back and forth all over memory, and access data at different locations, a smaller page size would speed up the program. A smaller page means that less data needs to be read from and written to the disk when pages are swapped.

If the user wishes to implement the whole simulated memory in main memory, he can by increasing the dimension of the array MEM to 65536. The page size should be set to 21512. With these variables set to the values indicated, the disk will never be referenced. It should be noted that array dimensions cannot usually exceed 32767 on most 16 bit machines. Another option the user has in eliminating the disk file, is to set the MEM array to a value less than 65536 and set the variable MXMEM to this dimension. Whenever an address exceeds the MXMEM value, an error message will be displayed.

5. The user may want to modify the standard display line associated with the "FORM S" command in order to display additional registers and status bits. This can be done by modifying the short display line write statement and its associated format statement. The variables of interest are listed below along with the Format by which they should be written to the output listing. The Write and Format statements are in Subroutine DISPL and are marked by comments.

Instruction Address	4A1	(IADDR(3,I),I=1,4)
Next Instruction Address	4A1	(IADDR(2,I),I=1,4)
Instruction Mnemonic	2A2	MNE1(ITYPE), MNE2(ITYPE)
Instruction Operand	10A1	(NOUT(I),I=1,10)
A Register	2A1	IROU1(8), IROU2(8)
B Register	2A1	IROU1(1), IROU2(1)
C Register	2A1	IROU1(2), IROU2(2)
D Register	2A1	IROU1(3), IROU2(3)
E Register	2A1	IROU1(4), IROU2(4)
H Register	2A1	IROU1(5), IROU2(5)
L Register	2A1	IROU1(6), IROU2(6)
Stack Pointer	4A1	(IADDR(1,I),I=1,4)
Cycle Count	4A1	(ICOUT(I),I=1,4)
Zero Flag	A1	FFO(1)
Carry Flag	A1	FFO(2)
Parity Flag	A1	FFO(3)
Sign Flag	A1	FFO(4)
Interdigit Carry	A1	FFO(5)
Effective (Operand) Address	4A1	(IADDR(4,I),I=1,4)
Effective Address Contents	2A1	IEA1,IEA2

6. The Simulator can recognize tab characters as field delimiters, but the uer must initialize the tab character in Subroutine INIT. Currently the tab character, NCTAB, is initialized to a blank in a DATA statement. The value that NCTAB must be initialized to varies from machine to machine. On many computers it is possible to encode the tab value as 1H(tab) in the DATA statement. If this is not possible on your machine, then the tab character will have to be initialized as a number. For most 16 bit ASCII machines this would be, NCTAB = 9*256+32. 9 is the ASCII value for a tab; 32 is the ASCII value for a blank. For PDP-11s, the bytes are switched, so NCTAB = 32*256+9. Most versions of NOVA Fortran do not have the trailing blank included so NCTAB = 9*256. Machines with word lengths greater than 16 bits must pad out the tab character value with as many blnaks as are in a host's word. Initializing a tab character will allow the Simulator to properly process an input source line that includes tabs. However, the Simulator does not expand the output line with tabs replaced by blanks. This must be done by the computer's operating system.

C. Batch/Interactive Mode

1. The program is delivered with the Batch/Interactive flag, IBAT, set to the interactive mode. In the Interactive mode, commands are not echoed to the listing device and errors do not cause program termination. In the Batch mode, commands are echoed to the listing device, all command errors cause program termination, and the command prompt is not displayed. Also in the Batch mode, messages are not displayed asking for input data. The user must determine the input data before executing the program and include it in the command stream.

D. Program Input/Output

1. The logical I/O device assignments made in the Simulator for the "Standard I/O Devices" are:

IRDR = 7 (object module input device)
IMFLE = 18 (intermediate file)

<u>Batch I/O</u>

ICRD = 5 (command input device)
IPRT = 6 (output listing device)

<u>Interactive I/O</u>

ITERC = 1 (command input device)
ITERP = 1 (output listing device)

These device assignments may have to be changed for your computer. This may be done either in the Job Control Stream that executes the Simulator or in the Program itself at compile time. If the assignments need to be changed in the program, the statements initializing the variables may be found in Subroutine INIT. Typically, the user only needs to change the Batch or Interactive assignments, since he will only be using the Simulator in one of these modes. Note that in the interactive mode both the input command device and the output listing device have the same device number. This is the usual case since they are typcially the same device.

2. When I/O is performed with a file, a logical device number is equated to the specified file so the file can be read from or written to by the I/O statements in Subroutine INOUT. The logical device numbers used for the various file types are shown below. After the file-logical device equating has been performed in Subroutine EQUAT, the file's logical device number is placed in a variable that represents the actual active input or output device. These are: IOCRD - command input device, IOPRT - output listing device, and IORDR - object module input device.

IFILC = 8 (command input logical device number which is equated to a file name)

IFILP = 9 (listing output logical device number which is equated to a file name)

IFILR = 10 (object module input logical device number which is equated to a file name)

It may be necessary to change these device assignments on your computer. The variables should be set to device numbers that can be equated with disk files.

3. The Simulator's intermediate file is a temporary file that is used to contain the microprocessor's simulated memory. This file must be random access. Some systems require disk space to be allocated for this temporary file in the Job Control Stream. Check to see if this is necessary for your computer. The Intermediate file is represented by the logical device number, IMFLE.

4. All Program I/O activity except for the generation of the output listing is handled in Subroutine INOUT. This includes displaying command prompts, reading the command input, and reading the input object module. Also included in INOUT is the display statements used for the SM, SIN, and SOUT commands. This display only occurs if the commands are continued on additional lines.

5. Reading and writing to a bulk storage device such as a disk is not standard in Fortran. There are however, two usual methods of performing these operations. Method 1 uses a DEFINE FILE statement and standard Read and Write statements as follows:

```
DEFINE FILE IMFLE(513,128,U,IMREC)
WRITE(IMFLE'IMREC) LIST
READ (IMFLE'IMREC) LIST
```

where: IMFLE - is the logical device number of the file
513 - is the maximum number of records in the file
128 - is the record length in words
U - indicates a binary record
IMREC - indicates the record number (associated variable)
LIST - list of variables to read or write

Method 2 uses a CALL to an executive or system routine to process the disk read or write. For a typical computer this call is:

```
CALL EXEC (#,CODE,IBUF,CNT,NAME,IMREC)
```

where: # - indicates the type of call, read or write
CODE - indicates binary or formatted I/O, etc.
IBUF - starting address of variables to read or write
NAME - is typcially a dimensioned array which contains the name of the disk file. This name is then used in the Job Control Stream to allocate the file.
IMREC - disk record number

The Simulator uses Method 1 as the standard method. However, statements for Method 2 are included in the program as comment statements for reference.

6. There are alternate ways of reading object modules and command files into the Simulator. They may be read from an I/O device (card reader, etc.) that can be refered to by a logical device number, or they may be read from disk files. If they are read from I/O devices that can be refered to by a logical device number, the number is used as the argument on the appropriate command. If the commands or object modules exist as disk files, the file name should be specified as the command argument. Subroutine EQUAT is used to equate a disk file and a logical device number so that the file may be read by the statements in Subroutine INOUT. The file logical device assignments for the various input and output devices are listed in 2. There are two basic parts to the EQUAT Subroutine. First, the file name is packed inot a contiguous Hollerith string. The code used to pack the characters into a string will work on any two's complement machine. For a one's complement machine, one line of code must be changed. The required change is marked with comments in Subroutine EQUAT. EQUAT also forms another array, IPBUF, which contains the file name in an A1 format, only one character per word. If the user must use the packed form of the file name to perform the acutal equate, two variables in Subroutine INIT must be set to the correct values for EQUAT to work properly. These are:

ISBIT - actual number of bits in host computer word. This may or may not be the same an IBIT.

ICHBT - number of bits per host computer character.

The place in INIT where these variables are set is marked with comments.

The second part of Subroutine EQUAT consists of the code required to open the named disk file and equate it to a logical device number. This code usually consists of one statement. Some computers can read disk files without any special code to open the file. In this case, Subroutine EQUAT may not be needed, or only needed to pack the file name into a contigious hollerith string. The array name, NAMEF, would then be placed in the file Read and Write statements in Subroutine INOUT. The user will have to check his computer manuals to find out what the required statements are to perform the file name and logical device number equate.

7. The I/O statements needed to read object modules and commands from disk files are usually the same as those that read from logical devices. However, they may be different and statements in INOUT provide for this case. The statements at 250 and 350 read Commands and Object modules from disk files. These statements are currently the same as those at 200 and 300 which read from logical devices. If the user must change the statements at 250 and 350, he can.

8. Statements are included in Subroutine INOUT for handling an End-Of-File (EOF) condition on both the command and object module input devices. The READ statements with the END condition specified, as shown in INOUT, can be used on most machines to detect an EOF. However, some systems require a call to a system routine or some other statement to detect an EOF. Comments in Subroutine INOUT are included to show where the program should branch when an EOF is detected. The user may not use the EOF feature to trace one instruction (see page 3-53) if an EOF cannot be detected. No modifications need to be made to the INOUT routine if EOFs cannot be detected. If the EOF conditions are not detected, the user should expect to get a system error if he does read through the end of a file.

9. In the Interactive mode, the program will display a prompt character (-) to request the next command from the user. If the user's system already displays a prompt for input data, the user may wish to remove the code that generates the Simulator's prompt. This code is located in Subroutine INOUT at Fortran statement number 100. When the prompt is displayed, users will probably not want the terminal to advance to the next line to read the command. Most systems have a format control that allows a line to be displayed on the terminal with no carriage return generated at the end of the line. For a NOVA, using the Z format is sufficient to prevent the carriage return. Thus, the format statement in Subroutine INOUT would be:

```
1000  FORMAT(1H-,Z)
```

For a PDP-11 the format statement would be:

```
1000  FORMAT(1H-,$)
```

10. The user may also want to inhibit a carriage return in the interactive mode when the SM, SIN, or SOUT commands are continued on additional lines. The statements displaying the address being modified for these commands are at Fortran statement number 400 in Subroutine INOUT. To prevent the carriage return from being generated for these commands Format statement 1003 should be modified in the manner described in 9.

11. A simplified EQUAT Subroutine for PDP-11 computers is shown below. This Subroutine may be used to replace the EQUAT Subroutine currently in the Simulator.

```
      LOGICAL*1  JNAME(18)

        REAL                  leave all statements down to
        INTEGER               Fortran Statement 100 from old EQUAT
        COMMON                Subroutine in new EQUAT Subroutine
          .
          .
          .
100   K = 1
110   IF(INC(JCOL).EQ.IBLNK) .OR. (INC(JCOL).EQ.ICOMM)) GO TO 200
      IF(INC(JCOL).EQ.ICTAB) GO TO 200
      IF(K .GT. 18) GO TO 920
      JNAME(K) = INC(JCOL)
      IPBUF(K) = INC(JCOL)
      K = K+1
      JCOL = JCOL+1
      GO TO 110
200   JNAME(K) = IBLNK
      IPBUF(K) = IBLNK
      CALL CLOSE(IFIL)
      CALL ASSIGN(IFIL,JNAME,0,'OLD')
      IDIV = IFIL
      IFIL = -IFIL
C     VALID RETURN
900   IERR = 0
      GO TO 990
C     FILE NOT FOUND
910   IERR = 1
      GO TO 990
C     ARGUMENT ERROR
920   IERR = 2
      GO TO 990
990   RETURN
      END
```

E. Memory Requirements and Overlays

1. If core size is limited, the Simulator program may have to be overlayed. One overlay structure is shown below. This overlay structure will have minimal effect on program speed.

Main	1st Overlay	2nd Overlay	3rd Overlay
MAIN	INIT	SIMU	LOAD8
INOUT		FUNC	EQUAT
COMMD			MESS
SIMU			
DISPL			
LABEL			
SYMBL			
SCAN			
MEMRW			
IORW			
AVHEX			

If necessary, additional routines can be placed in the 3rd Overlay. However, program speed may be noticeably affected.

2. To aid those users who need to form their own Overlays or to Segment their programs, the following list shows each routine in the Simulator and all the routines that call it.

MAIN	-	
INIT	-	
INOUT	-	COMMD,LOAD8,MEMRW,IORW
COMMD	-	MAIN
SIMU	-	COMMD
DISPL	-	COMMD
LOAD8	-	COMMD
LABEL	-	COMMD,SCAN
SYMBL	-	LOAD8,LABEL
SCAN	-	COMMD,LOAD8,IORW,EQUAT
MEMRW	-	COMMD,SIMU,DISPL,LOAD8
IORW	-	SIMU,MEMRW
FUNC	-	SIMU
AVHEX	-	COMMD,DISPL,IORW
EQUAT	-	COMMD
MESS	-	COMMD,LOAD8,IORW

The following list lists each Subroutine in the Program and the routines it calls.

```
MAIN  - INIT,COMMD
INIT  -
INOUT -
COMMD - INOUT,LABEL,LOAD8,MEMRW,SCAN,AVHEX,DISPL,EQUAT,MESS
SIMU  - FUNC,IORW,MEMRW
DISPL - AVHEX,MEMRW
LOAD8 - INOUT,MEMRW,MESS,SCAN,SYMBL
LABEL - SYMBL
SYMBL -
SCAN  - LABEL
MEMRW - INOUT,IORW
IORW  - INOUT,AVHEX,MESS,SCAN
FUND  -
AVHEX -
EQUAT -
MESS  -
```

3. If the user cannot or does not want to create overlays, there are three things he can do to reduce the size of the program.

   A. Currently the Error Message Subroutine, MESS, writes out English messages to the listing. This routine could be replaced with a simple routine that contained one write statement that wrote out the error message number, MESSN. The user would then refer to a listing of the old MESS routine to find out what the error number indicated.

   B. Eliminate or reduce the size of the symbol table (see Section A.3).

   C. Reduce the page size of the simulated memory page to 256 or 128 words (see Section A.4).

F. NOVA Modifications

When installing the Simulator on a NOVA Computer, it is suggested that Fortran V be used. If Fortran IV is used, some additional program modifications have to be made.

1. Most versions of NOVA Fortran fill an H data specification statement with zeros and not blanks, as is typcially done. Therefore, characters read in under A formats must have the padded blanks stripped off so they will match equivalent characters stored in the program under H formats. Insert the following statements after Fortran statement 380 in INOUT.

```
      DO 382 I=1,80
      IN(I) = IN(I).AND-256
382   CONTINUE
```

2. All variables initialized in DATA statements must be placed in Labeled COMMON. The variables are local to each Subroutine, so unique dummy labels may be used for the COMMON Block names.

3. The DEFINE FILE Statement in the Main Program must be replaced with a CALL OPEN statement similar to the one shown below.

```
CALL OPEN(IMFLE,"IDUM1",3,IER,256)
```

4. The Simulator intermediate file must be a random access file, so a Call to FSEEK must preceed each file access. Use Binary Read and Write statements for the intermediate file. To implement this, change the Fortran source code in INOUT as follows:

```
    CALL FSEEK (IMFLE,IMREC)
    IF(ICTL .EQ. 7) GO TO 630
    READ BINARY  (IMFLE) (MEM(I),I=I1,I2)
    GO TO 640
630 WRITE BINARY (IMFLE) (MEM(I),I=I1,I2)
640 I1 + I2+1
```

5. Several characters cannot be used in Hollerith Data Specifications since they are not in the NOVA assembler's legal character set. These include right and left parenthesis, percent sign, quote mark, etc. Check your Assembly Language Manual for the legal character set. In Subroutine INIT replace all illegal characters in the array NALPH with their internal representations as they would appear in a 1H Data format.

```
  DATA NALPH(37),NALPH(38),NALPH(39),NALPH(40)  /16128,1H@,1H ,1H!/
  DATA NALPH(41),NALPH(42),NALPH(43),NALPH(44)  /8704,1H#,9216,9472/
  DATA NALPH(45),NALPH(46),NALPH(47),NALPH(48)  /1H&,9984,10240,10496/
  DATA NALPH(57),NALPH(58),NALPH(59)            /15360,1H=,15872/
```