SIMULATOR AND DEBUGGER FOR INTEL-8080

January 1977

Alf-Martin Jensen

Table of Contents

1 Introduction

The INTEL-8080 simulator and debugger for the NORD-1 and NORD-10 computers is designed to be used for testing and debugging programs written for the INTEL-8080 microprocessor.

The debugger has facilities for examining and changing the contents of memory, central registers and other variables that are built into the system. All registers have mnemonic names by which they may be accessed. Memory locations may also be examined and changed in symbolic format (i.e. as instructions). Program input to the simulator is in standard INTEL hexadecimal format. If the symbol table was output to the object file at assembly time, it is automatically loaded. This allows the user the convenience of symbolic references to memory locations, to a large degree eliminating the constant need for an up-to-date assembly listing with absolute addresses. A particularly powerful debugging tool is the conditional breakpoint facility which allows the user to specify the exact condition under which he wants program execution to terminate. Machine code is compiled for the breakpoint expressions, so that even a complicated condition will not result in an increase in execution time of more than 5-10%. Another powerful feature is the program counter list. This is a list that always contains the last 64 values of the program counter, making it possible to trace program execution backwards in time.

The 8080 memory area is allocated in one of two different ways: in main memory or on a core image file. When allocated in main memory the address space is limited to the size of the memory buffer, but program execution is considerably faster than if it is allocated on a file. The size of the memory buffer is determined at system generation time. If memory is allocated on a core image file, the whole 16 bit address space is available, but execution speed depends upon the behaviour of the program and the number of file buffers available. Two debugger commands makes it possible to change the mode of operation whenever the user wishes to.

The I/O system makes it possible to simulate both isolated and memory mapped I/O. Any set of I/O addresses may be attached to any file, and the user can specify the driver routine (written in NORD-10 code) which is to perform the I/O operation. The driver routine is identified by its absolute (NORD-10) address. A standard calling sequence is specified for driver routines. This means that the user can write his own driver routines and link them to the simulator. Standard driver routines which perform binary and ASCII format I/O are built-in.

The program occupies about 11 K words of memory plus memory buffer, file buffers and stack area. It runs under the operating systems NORD-TSS and SINTRAN-III without modification.

2 Expressions

2.1 Constants

A constant is a string of digits and letters. The first character must be numeric. If the last character of the string is B, O, Q, D, H or S it is used as a radix specifier. The letters have the following meaning: B, binary; O or Q, octal; D, decimal; H, hexadecimal; S, split octal. If no radix specifier is present, the number is assumed to be represented in the default radix. The default radix is normally 16 (hexadecimal), but may be changed by means of the ;C command as described below. The number represented by the string is evaluated and truncated to 16 bits.

2.2 String Constants

A string constant is an arbitrary string of ASCII characters, other than carriage return, enclosed within single quotes. If a single quote is to be included in the string, it must be written as two single quotes. The maximum length of a string constant is 80 characters.

2.3 Symbols

A symbol is a string of letters and digits. The first character must be alphabetic or the special character "?". In symbols consisting of more than five characters, only the first five are significant: thus, SYMBOL is equivalent to SYMBO. Two kinds of symbols are used in the debugger: predefined symbols (e.g. register names) and variable symbols. The latter are either loaded from the object input file or assigned values by means of the assign command. Predefined symbols are preceeded by a dot (".") and variable symbols by a dollar sign ("$"). As an example .B refers to the B-register of the 8080 CPU while $B refers to the variable symbol B. To ease the use of symbols the type specifier ("." or "$") may be omitted as long as the symbol type can be determined from the symbol name. For example, the symbol B usually refers to the B-register, but if a variable symbol named B is defined one must write .B (and $B). In addition, all the operation codes are built-in symbols, but they may only be used when typing in expressions in symbolic form as explained in a later section. Following is a list of all predefined symbols together with an explanation of their meaning.

Symbol Description

B	B-register (8 bits)
C	C-register (8 bits)
D	D-register (8 bits)
E	E-register (8 bits)
H	H-register (8 bits)
L	L-register (8 bits)
A	Accumulator (8 bits)
BC	Double register BC (16 bits)
DE	Double register DE (16 bits)
HL	Double register HL (16 bits)
P	Program counter (16 bits)
SP	Stack pointer (16 bits)

CC Condition codes (8 bits)
bit 0 = carry
bit 2 = parity
bit 4 = auxiliary carry
bit 6 = zero
bit 7 = sign

IR Instruction register (8 bits)

I Instruction counter (15 bits). The instruction counter is updated only if the maximum instruction counter (MI) is non-zero.

MI Maximum instruction count (15 bits). When zero the instruction counter (I) is not to be updated. Otherwise execution terminates when I becomes equal to MI.

SBASE The value of this symbol is the smallest address which the debugger will ever attempt to print in symbolic form.

MDISP This is the maximum displacement that will be printed when an address is typed out in symbolic form. If the displacement is greater than this vaue, the address is printed in constant form.

The symbol "." refers to the contents of the last opened location. The contents of a register or a memory location is specified by enclosing it in brackets. For example [129] denotes the contents of memory location 129 while [P] or [.P] denotes the contents of the P-register.

2.4 Simple Expressions and Breakpoint Expressions

Two kinds of expressions are used in the debugger: simple expressions for use when examining and changing locations etc. and breakpoint expressions for use when specifying breakpoint conditions. Breakpoint expressions are used only with the ;B command and are logical expressions which return the value 0 (FALSE) or -1 (TRUE). Expressions are built up from operators and operands. Operands may themselves be expressions if enclosed within parentheses. The operators, in order of decreasing priority, are listed below for both types of expressions.

Simple Expressions

Priority	Operation	Description
1	*	Multiplication
2	+	Addition
2	-	Subtraction
3	NOT	One's Complement
4	AND	Arithmetic And
5	OR	Arithmetic Or
5	XOR	Arithmetic Exclusive Or

Breakpoint Expressions

Priority	Operation	Description
1	*	Multiplication
2	+	Addition
2	-	Subtraction
3	NOT	One's Complement
4	AND	Arithmetic And
5	OR	Arithmetic Or
5	XOR	Arithmetic Exclusive Or
6	<<=	Magnitude less than or equal
6	<<	Magnitude less than
6	<=	Less than or equal
6	<>	Not equal
6	<	Less than
6	=	Equal
6	>>=	Magnitude greater than or equal
6	>=	Greater than or equal
6	>>	Magnitude greater than
6	>	Greater than
7	.N	Logical Not
8	&	Logical And
9	!	Logical Or

The operators with priority 6 are relational operators that return either 0 (FALSE) or -1 (TRUE). The operators .N, & and ! can only be used with operands of this type. For example, [.P] = 100 & [.BC] <> 3 is a valid breakpoint expression while [.P] =100 & 3 is not. The magnitude tests regard their operands as 16 bit positive numbers while the other tests regard them as 16 bit two's complement numbers.

2.5 Instructions

Instructions can be typed in to the debugger in symbolic form. All operation codes that represent machine instructions (no directives) are recognized. Register operands can only be simple registers or register pair names, while addresses and immediate operands can be simple expressions.

2.6 Formal Definition of The Syntax of Expressions

The following is a formal definition of the syntax of expressions. The symbol "/" means or and "$" means arbitrary number of. Brackets enclose optional items, parentheses enclose groups of items and double quotes enclose literals. B:expr means breakpoint expression and s:expr means simple expression.

```
s:expr=     conj $(("OR"/ "XOR") conj);
conj=       neg $("AND" neg);
neg=        ["NOT"] sum;
sum=        term $(("+"/ "-") term);
term=       prim $("*" prim);
prim=       "(" s:expr ")"/
            "[" regsym/ s:expr "]"/
            ["+"/ "-"] (const/ symbol)/ ".";

b:expr=     l:alt $("!" l:alt);
l:alt=      l:neg $("&" l:neg);
l:neg=      [".N"] rel;
rel=        "(" b:expr ")"/
            a:expr relop a:expr;
relop=      "<<="/ "<<"/ "<="/ "<>"/ "<"/
            "="/ ">>="/ ">="/ ">>"/ ">";
a:expr=     a:alt $(("OR"/ "XOR") a:alt);
a:alt=      a:neg $("AND" a:neg);
a:neg=      ["NOT"] a:sum
a:sum=      a:term $(("+"/ "-") a:term);
a:term=     a:prim $("*" a:prim);
a:prim=     "(" a:expr ")"/
            "[" regsym/ a:expr "]"/
            ["+"/ "="] (const/ symbol);
regsym=     <any of the symbols defined in section 2>;
symbol=     ("?"/ alpha) $4 (alpha/ num);
const=      b:const/ o:const/ d:const/
            h:const/ a:const/ s:const;
b:const=    1$ ("0"/ "1") "B";
o:const=    1$ o:digit ("O"/ "Q");
d:const=    1$ num ["D"];
h:const=    1$ (num/ "A"/ "B"/...../ "F") "H";
a:const=    "'" 1$2 (<any character>-"'"/ "'"'") "'";
s:const=    [byte] byte "S";
byte=       [["0"/ "1"/ "2"/ "3")] o:digit] o:digit;
o:digit=    "0"/ "1"/...../"7";
alpha=      "A"/ "B"/...../"Z";
num=        "0"/ "1"/...../"9";
```

3 Commands

This section describes all the available debugger commands. The symbols a, b and c denotes simple expressions, cr means carriage return and lf means line feed. Addresses may be printed in one of two different formats: as a constant represented in the default radix or as a symbol name (only labels) plus a displacement. Two parameters control the symbolic printout format: the predefined symbols SBASE and MDISP. SBASE is the smallest address which the debugger will attempt to convert to symbolic form. MDISP is the maximum displacement that will be printed. If the displacement is greater than this value, the address is printed in constant form. SBASE and MDISP are initialized to zero and 177Q respectively, but may be changed by the user. It is assumed that addresses are printed in constant form when not otherwise stated. In the examples the default radix is assumed to be 16 and user input is underlined.

Command	Description
expression :	Display value of 'expression' in the default radix. Examples: 2+4: 6 (2+5)*10B: E
expression /	Examine item specified by 'expression'. The contents of the item is displayed as a constant represented in the default radix. If expression is null, the last opened item is reopened. Examples: P/ 003A 3A/ C6 / C6
expression \	Examine memory location specified by 'expression' in symbolic form. If the memory location contains a legal instruction it is disassembled and typed out. Otherwise it is printed as a constant represented in the default radix. Addresses and 16-bit immediate operands are converted to symbolic form. If expression is null, the last opened location is reopened. examples: 23\ LDA BUFF+2 256\ MOV B,C
expression "	Examine string. The memory location specified by 'expression' and the following locations are typed out as an ASCII string. The string is terminated by any non-printable character other than carriage return or line feed. In order to avoid large

amounts of output if the locations do not contain a string terminated in the manner described above, typeout is terminated when 72 characters without any intervening carriage returns have been printed.

expression, a "

This command is similar to the command described above, except that the value of a specifies the terminating character. examples:

STR" THIS STRING IS TYPED OUT
STR,'Y'" THIS STRING IS T

line feed

Examine next sequential item. The output produced by this command is carriage return, location or name of item,/, contents of item. If the opened item was a memory location, the address is printed in constant form if it was opende by / and in symbolic form if it was opened by \. If the last item was printed in symbolic form, the address is incremented by the length of the printed instruction (1, 2 or 3 bytes). Examples:

B/ 06 1f
.C/ 08
3A/ 06 1f
003B/ 03 1f
003C/ 0C
3A\ MVI B,03 1f
003C\ NOP

↑

Examine previous item. This command is similar to line feed but is illegal if the item was opened by \. Examples:

3A/ 06 ↑
39/ 05

x terminator

Deposit value. After an item has been opened with /, \, ", ^ or line feed, a new value may be deposited by typing an expression (x) followed by a terminator. The item to be deposited may be a simple expression, a string constant or an instruction in symbolic format. The terminator may be carriage return, line feed or ^. Examples:

P/ 0317 3+5 cr
P/ 0008
34/ 08 9 1f
35/ 17 ↑
34/ 09

3A/ 06 MVI C,4 If
003C\ NOP
3A\ MVI C,C4

reference = expresion cr — Assignment. The value of the expression on the right side of the equal sign is assigned to the reference on the left. The reference can be a variable symbol or a reference to a register or a memory location. Examples:

ABC=3 (or $ABC=3), assign the value 3 to the symbol ABC. [P]=5 (or [.P]=5), set the P-register to 5. [[P]+2]=[2], set the memory location addressed by the P-register plus two to the contents of memory location 2.

@ string cr — Execute SINTRAN-III command. The string is sent to the SINTRAN-III comand processor for execution.

;@ — Return to monitor.

;A — Examine all I/O assignments.

interval;A a, file cr — I/O assignment. The specified file is attached to all I/O devices with addresses in the specified interval. The I/O operation is to be performed by a driver routine located at the absolute (NORD-10) address a. If a is less than 64 it specifies one of the built-in driver routines. Read the section describing the I/O system for further information.

;B — Reset all breakpoints currently defined.

-a;B — Reset breakpoint a (1<=a<=63).

a;B string cr — Define breakpoint a (1<=a<=63). The string must contain a valid breakpoint expression. The expression may be followed by a comment string which consists of a semicolon followed by an arbitrary string. The simulator will stop executing instructions whenever the value of any breakpoint expression becomes TRUE. Each breakpoint is tested before every instruction.

;C — Change default radix to 16.

a;C — Change default radix to a (2<=a<=36).

0;C — Change default radix to split octal.

;D	Dump register block. The registers in the register block, the stack pointer, the condition codes and the contents of the memory location currently addressed by HL are printed on the terminal.
;E	Examine all breakpoints currently defined.
a;E	Examine breakpoint a (1<=a<=63).
;G	Start 8080 CPU at the current value of the program counter.
a;G	Start 8080 CPU at the address specified by a.
;J	Delete all I/O assignments.
interval;J	Delete the I/O asignment specified by 'interval'. The exact interval must be specified. Read the section describing the I/O system for further information.
;K	Clear the debugger's variable symbol table.
s1,s2,..,sn;K	Kill symbols. The symbols s1,s2,..,sn are removed from the debugger's variable symbol table.
;L	List the program counter list. The simulator saves the addresses of executed instructions in a 64 word circular list. The ;L command prints these addresses on the terminal. The addresses of the most recently executed instructions are displayed first. This facility is useful for tracing a program backwards in time.
a;L	List a entries in the program counter list.
0;L	Clear the program counter list.
a,b;N	This command prints the contents of the memory locations between the addresses a and b in symbolic format. The addresses of the locations are printed in symbolic form.
;O file name cr	Open core image file. This command specifies that 8080 memory resides on a core image file. The entire address space (16 bits) is available for program and data storage. The default file type of the file is :CORE.

;P — Proceed with execution (after a break). This command is similar to ;G except that the instruction counter and status information related to breakpoints are not reset.

;Q — This command specifies that 8080 memory resides in a memory buffer. The available address space is determined at system generation time. The currently open core image file is closed. The debugger is initially in this mode.

;R — Reset simulator and debugger. Default radix is set to 16, the variable symbol table is cleared, the program counter list is cleared, all breakpoints are reset, the instruction counter and the maximum instruction counter are set to zero, SBASE is set to zero, MDISP is set to 177Q, and the program counter (P-register) is set to zero.

;S — Execute one instruction.

a;S — Execute a instructions, breakpoints are ignored.

a,b;S — Execute a instructions starting at the address specified by b. Break points are ignored.

;T — Print all symbols in the variable symbol table on the terminal.

;U — Output generator. When ;U is typed, the following parameters are asked for:

FORMAT (HEX, BPNF)
INPUT FILE
OUTPUT FILE
LOWER BOUND
UPPER BOUND

and when finished:

MORE (Y OR N)

If an input file is specified, the contents of this file is read into memory, otherwise the contents of memory is not altered. The input file must contain code in hexadecimal format and the default file type is :HEX. The output file will contain the memory area specified by the lower and upper bounds, represented in the desired format. The default

file type of the output file is :BPNF if the output format is BPNF and :HEX if the output format is hexadecimal. If the question MORE is answered by Y, the lower and upper bounds are asked for again and more output may be written onto the same file. If hexadecimal format is specified the symbols in the variable symbol table are also output.

0;U

This command is similar to ;U except that the symbol table is not output if hexadecimal format is specified.

a,b;W cr

This command prints the contents of the memory locations between the limits a and b. If an area of length greater than or equal to four contains the same value, it is printed in a compressed format. For example, if location 300H and upwards contais 10H, 20H, 0, 0, 0, 0, 0, 30H, 40H etc. the command 300H,308H;W will produce the following output.

```
0300 10
0301 20
0302 00
   .
   .
   .
0306 00
0307 30
0308 40
```

a,b;W c cr

This command searches memory between the limits a and b for locations whose contents is c. The address and contents of all such locations are typed out. The format is the same as described above.

;Y file name cr

Load program. The specified file must contain code in hexadecimal format. The contents of this file is loaded into memory. The symbol table (if any) is also loaded. Symbols that already exists in the debugger's variable symbol table are redefined, while new entries are created for new symbols. The default file type is :HEX.

a,b;Z cr

This command sets to zero all memory locations in the interval a through b.

a,b;Z c cr

This command sets all memory locations in the interval a through b to the value of c.

4 The I/O System

4.1 The I/O System Commands

I/O assignments are specified with the ;A command which has the following format:

address interval;A driver routine, file name

The address interval consists of one expression or two expressions separated by comma, the former specifying an interval that consists of only one address. The first expression must be preceeded by a prefix that specifies the type of I/O address. If the second expression is preceeded by a prefix it must be the same as the first one. The legal prefixes are:

I	Isolated I/O, input (IN instruction)
O	Isolated I/O, output (OUT instruction)
MI	Memory mapped I/O, input (load instructions)
MO	Memory mapped I/O, output (store instructions)

Intervals with different prefixes can overlap while those with the same prefix can not. If the specified interval is already assigned, its driver routine and file assignments are changed to the new value.

The driver routine is identified by its absolute (NORD-10) address. If no driver routine, or zero, is specified, the current value of the address is not changed. If no driver routine, or zero, is specified and the interval is a new one, the ASCII format I/O driver routine is assigned by default. If the address is less than 64 it specifies one of the built-in driver routines. The built-in driver routines and the calling sequence of driver routines are described below.

The file name specifies a file that can be used in I/O operations on devices with addresses in the specified interval.

The I/O assignments may be examined by typing ;A without any arguments. The I/O table is printed on the terminal in the following format:

prefix low address, high address file number driver routine

The I/O addresses are printed in the default radix while the file number and the address of the driver routine are always printed in octal.

To cancel an I/O assignment the command interval;J is used. The exact interval must be specified. ;J without any arguments cancels all I/O assignments.

The I/O table is not initialized when the simulator is started. This makes it possible to create a system with permanent assignments by executing the SINTRAN-III command DUMP. The start and restart addresses are specified on the system tape.

If an IN or OUT instruction, for which no I/O assignment exists, is executed, the ASCII format driver routine is used.

4.2 Built-in Driver Routines

Two built-in driver routines exist. These are specified by 1 for ASCII format I/O and 2 for binary format.

4.2.1 ASCII Format I/O

When the simulator encounters an input instruction the action taken depends on whether or not the specified file is the terminal. If the file is the terminal the driver routine outputs the current value of the program counter and the I/O address preceeded by a prefix. It then requests input from the user. For example if the instruction 'IN 23H' is stored in location 303H, the following output is produced:

```
P=0303  I 0023=
```

The user is now expected to type the correct input value, which may be an expression as defined in section 3. If ctrl-L is typed, program execution terminates and control returns to the debugger.

If the file is any file other than the terminal the driver routine reads the next item from a list of values stored on this file. These values may be expressions and they may be separated by space, comma or carriage return. As an example the following is a valid list of values:

```
17H 23Q, 13Q+25D cr
19H 2, 5 cr
```

If an error occurs or if end-of-file is reached, an error message is printed and control returns to the debugger.

When an output instruction is encountered the driver routine outputs the current value of the program counter, the I/O address preceeded by a prefix and the output value. For example if the instruction 'OUT 46H' is stored in location 902H and the A-register contains 22H the following is written onto the assigned file:

```
P=0902  O 0046=22
```

The default file for ASCII format I/O is the terminal.

4.2.2 Binary Format I/O

In binary mode bytes are transferred directly from the A-register to the file or from the file to the A-register. There is no default file for binary I/O.

4.3 Calling Sequence of Driver Routines

At entry:

```
T =     bit 15 = 0 if isolated, 1 if memory
        bit 14 = 0 if input, 1 if output
        bits 7-0 = file number
                 = 0 if no file assigned
A =     data if output operation
D =     pointer to a location which contains
        the address of the memory access routine
X =     the I/O address
```

Return: (failure)

```
A =     0 if return to debugger
        <>0 if print message addressed by A
            and return to debugger
```

Skip return: (success)

```
A =     data if input operation
```

The calling sequence of the memory access routine:

At entry:

```
T =     0 if read, 1 if write
A =     data if write
X =     memory address
```

Return: (failure)

Skip return: (success)

```
A =     data if read
```

5 Operating Instructions

To start the simulator from NORD-TSS or SINTRAN-III one types the following.

@SIM-8080 cr

INTEL-8080 SIMULATOR V01.02

The debugger is now ready to accept commands. To return to the monitor type :@.