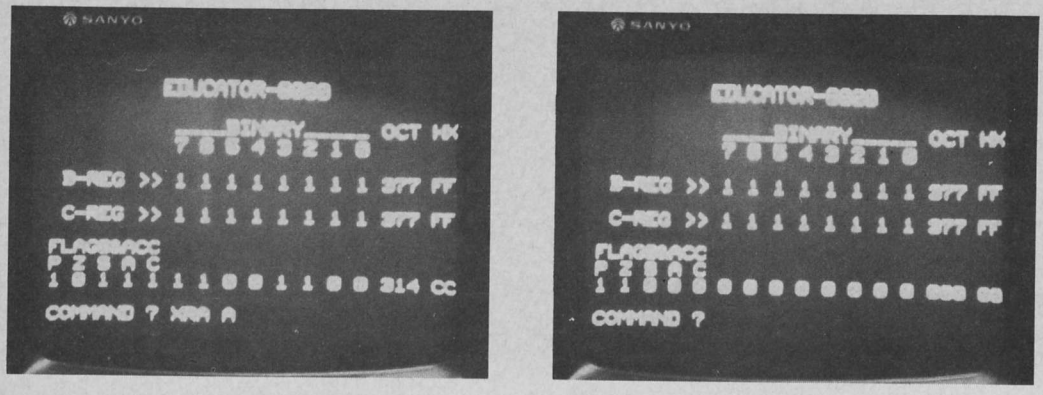


Photo 1: In this example, Educator-8080 is shown before (left) and after (right) the execution of an XRA A instruction. The effect of this instruction is of course to clear the accumulator A, as is shown at the right.

Explore an 8080 with Educator-8080

Charles P Howerton
President, Digital Group Software Systems Inc
PO Box 1086
Arvada CO 80001

What Is an Educator?

Educator-8080 was designed as a classroom instruction aid for a microprocessor programming course. The principal design goals were to develop a system which would illustrate the architecture of the machine and the effect of the execution of various instructions. For example, the reader might ask to what use the logical EXCLUSIVE OR function may be put in an 8080. This function, which operates on each bit, has a value of 1 if either of the two operands or arguments, but not both, has a value of 1; otherwise it has a value of 0. The Educator-8080 can simply illustrate this function. In the example shown in photo 1 (left and right), both arguments of this function are equal: the first argument is the value in the accumulator (A) and the second argument is also the value stored in the accumulator (A). The function value (ie: the result) is placed in the accumulator after execution. The left photo shows the accumulator (and other registers, etc, which are not affected) before execution of the instruction XRA A, and the right photo shows it after execution. The result is that the accumulator (A) is cleared, ie: it contains eight 0 bits. This result is consistent with the definition of the EXCLUSIVE OR above: Whenever both bit arguments are 0, or are 1, a value of 0 is returned. This example shows that the Educator-8080 is a convenient means to illustrate rather complex operations which facilitate learning the instruction set and architecture without the tedium of plowing through books. A subordinate goal was to implement the entire system with the exception of the physical input output routines and the stack in 1024 bytes of memory. All of the design goals were met. In addition, if the IO devices are ASCII oriented, a reduction in the length of the error messages (perhaps limiting them to the error code number) should provide sufficient space for the inclusion of the physical input output routines and the stack within the 1 K byte memory space.

Educator-8080 is written in a fairly straightforward manner and it should not be particularly difficult to adapt it to any 8080 system with more than 1 K bytes of pro-

grammable memory, a keyboard, and an output device of some kind. It is designed to operate with a television display device and to dynamically show the results of the execution of the input commands.

It would probably be desirable to modify the display function somewhat if the output device is a hard copy device such as a Teletype. The content could be the same but the elimination of blank lines and printing the titles only every 10 or 15 instructions would speed things up considerably on a Teletype device. The input output routines required to adapt Educator-8080 to almost any system are described functionally but are not given in detail. They should be adapted from routines already in use for a given system.

The Instruction Set

The Educator-8080 instruction set is a subset of the 8080 instruction set. The commands implemented within Educator-8080 were selected to provide representative instructions from most of the functional instruction groups. Since the instructions are to be executed one at a time from keyboard input, there was no need to incorporate any of the Jump, Call or Return instructions; however, since the flags are displayed after each operation, it is very easy to determine whether or not a given conditional Jump, Call or Return would cause a transfer of control by simply observing the setting for the flag whose status is being tested. In addition, because of memory limitations none of the instructions which cause memory to be read or written were implemented. Finally, no instruction whose action could not be readily observed was implemented.

To keep the display as uncluttered as possible the registers which could be accessed by Educator-8080 instructions were limited to the accumulator and the B and C registers. It would not be particularly difficult to incorporate the rest of the registers into the display and as operands for the Educator-8080 instruction subset. However, unless the ability to address memory is desirable the only instructions which could be added to the subset would be the DAD and the XCHG.

The instruction subset and the valid operands for each instruction are shown in table 1. Table 2 contains the corresponding information as it is loaded into the computer's memory and used by Educator-8080.

Immediate Operands

Almost half of the instructions supported by Educator-8080 require immediate

Table 1: Command List for Educator-8080. In order to illustrate the operations of the 8080 processor, Educator-8080 interprets a subset of the 8080's instructions. The subset generally references the accumulator, A, and registers B or C; it excludes all branching and program control operations. The complete list of available operations is found in this table.

Command	Description of Operation	—Flags— P	Z	S	A	C
ACI i	Add the value of the **Carry Flag** and the value of the immediate operand **i** to the contents of the **accumulator**.	X	X	X	X	X
ADC r	Add the value of the **Carry Flag** and the contents of register **r** to the contents of the **accumulator**.	X	X	X	X	X
ADD r	Add the contents of register **r** to the **accumulator**.	X	X	X	X	X
ADI i	Add the value of the immediate operand **i** to the **accumulator**.	X	X	X	X	X
ANA r	Logically AND the contents of register **r** with the **accumulator**.	X	X	X	X	0
ANI i	Logically-AND the value of the immediate operand **i** with the contents of the **accumulator**.	X	X	X	X	0
CMA	Complement the contents of the **accumulator**, changing all of the zeros to ones and all of the ones to zeros.	N	N	N	N	N
CMC	Complement the value of the **Carry Flag**; if it is zero make it one, or if it is one make it zero.	N	N	N	N	X
CMP r	Compare the contents of register **r** with the contents of the **accumulator**.	X	X	X	X	X
CPI i	Compare the value of the immediate operand **i** with the contents of the **accumulator**.	X	X	X	X	X
DAA	Decimal adjust the value in the **accumulator** (after an arithmetic command using decimal numbers).	X	X	X	X	X
DCR r	Decrement (subtract 1 from) the contents of register **r**	X	X	X	X	N
DCX rp	Decrement the contents of the register pair **rp**.	N	N	N	N	N
INR r	Increment (add 1 to) the contents of register **r**.	X	X	X	X	N
INX rp	Increment the contents of the register pair **rp**.	N	N	N	N	N
MVI r, i	Move the value of the immediate operand **i** into register **r**.	N	N	N	N	N
MOV r, s	Move the value of the contents of register **s** into register **r** leaving **s** unchanged.	N	N	N	N	N
NOP	No operation: do nothing.	N	N	N	N	N
ORA r	Logically OR the contents of register **r** with the **accumulator**.	X	X	X	0	0
ORI i	Logically OR the value of the immediate operand **i** with the **accumulator**.	X	X	X	0	0
RAL	Rotate the contents of the **accumulator** left one bit position with the high order bit going to the **Carry Flag** and the **Carry Flag** going into the low order bit of the **accumulator**.	N	N	N	N	X
RAR	Rotate the contents of the **accumulator** right one bit position with the low order bit of the **accumulator** going into the **Carry Flag** and the **Carry Flag** going into the high order bit of the **accumulator**.	N	N	N	N	X
RLC	Rotate the contents of the **accumulator** left one bit position with the high order bit going into both the low order bit and the **Carry Flag**.	N	N	N	N	X
RRC	Rotate the contents of the **accumulator** right one bit position with the low order bit going into both the high order bit and the **Carry Flag**.	N	N	N	N	X
SBB r	Subtract the values of the **Carry Flag** and register **r** from the **accumulator**.	X	X	X	X	X
SBI i	Subtract the values of the **Carry Flag** and the immediate operand **i** from the **accumulator**.	X	X	X	X	X
STC	Set the **Carry Flag** to a 1 value.	N	N	N	N	1
SUB r	Subtract the contents of register **r** from the **accumulator**.	X	X	X	X	X
SUI i	Subtract the value of the immediate operand **i** from **accumulator**.	X	X	X	X	X
XRA r	Logically Exclusive OR the contents of register **r** with the **accumulator**.	X	X	X	0	0
XRI i	Logically Exclusive OR the value of the immediate operand **i** with the **accumulator**.	X	X	X	0	0

Key:
i = any valid immediate operand (see text).
r = any one of the three registers displayed A, B, or C.
rp = must be the register pair B and C which is designated B.
s = any one of the three registers displayed A, B, or C

Values for FLAGS:
X = Changed value depends on operands and command.
0 = Reset to zero always.
1 = Set to one always.
N = Not changed by this command.

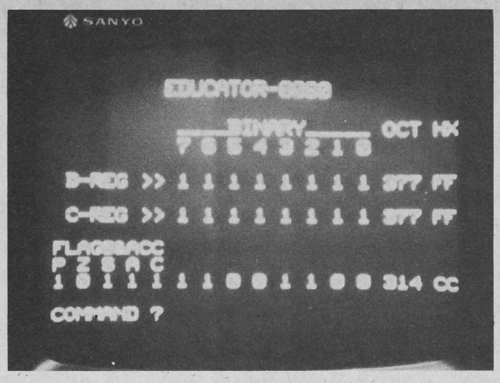


Photo 2: What happens when an 8080 executes an ADD B instruction? A specific example is illustrated in this set of before and after snapshots.

operands. An immediate operand is a constant value which is part of the instruction being executed and it immediately follows the operation code of the instruction, hence the name immediate.

Whenever a single byte "constant" is required in a program, its inclusion as the immediate value of an appropriate instruction reduces the length of the program because there is no need to address the value directly. Immediate values have an implied address which is the address of the byte following the opcode and this address is supplied from the program counter register automatically whenever an immediate type instruction is executed. In the Educator-8080 system the "program counter" is provided by the operator's sequence of commands which are executed one by one.

Educator-8080 has three different types of immediate values as part of the input command and defaults to one of these types if the input command omits type information.

The general form of an immediate operand is as follows:

T p V p

Where: T is the type code which designates the form of the immediate value and may be any of the following:

B – for a binary immediate value

Q – for an octal immediate value

H – for a hexadecimal immediate value

If the type code is omitted entirely and the first nonpunctuation character encountered is a numeric digit 0 to 7, then a default type of octal is assumed.

p is any form of punctuation (eg: single or double quotes, parentheses, etc). Punctuation is not required, and provision for its inclusion is solely in the interest of enabling the user to enter commands in a format consistent with that of various advanced assemblers.

V is the value of the immediate operand expressed in a form consistent with the explicit or implied type selected. The form and content of the value field for each type is as follows:

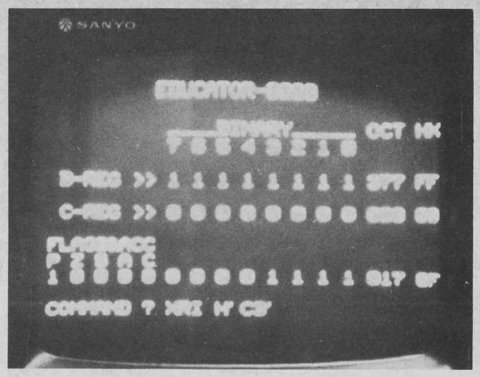
T = B: V is a series of eight consecutive numeric characters which have the value zero or one.

Example: **B'11000111'**. V is **11000111**, quotes are optional.

T = Q or T omitted: V is a series of

Table 2: Operation Code Table for the Educator-8080 program. Table 1 showed the command list for the program. This table gives the absolute machine codes for the command table beginning at address <2>/122. Each command is represented by a 3 byte ASCII character string mnemonic followed by the naked (without register values) 8080 operation code and the address of the routine which interprets the command. The routine name is shown symbolically in the right hand column, and can be found in the program of listing 1.

Address	ASCII Mnemonic	Octal Code Mnemonic	Opcode	Routine	Routine Name
<2>/122	'ACI'	101 103 111	316	212<1>	IMMED
<2>/130	'ADC'	101 104 103	210	152<1>	RG210
<2>/136	'ADD'	101 104 104	200	152<1>	RG210
<2>/144	'ADI'	101 104 111	306	212<1>	IMMED
<2>/152	'ANA'	101 116 101	240	152<1>	RG210
<2>/160	'ANI'	101 116 111	346	212<1>	IMMED
<2>/166	'CMA'	103 115 101	057	144<1>	DIRCT
<2>/174	'CMC'	103 115 103	077	144<1>	DIRCT
<2>/202	'CMP'	103 115 120	270	152<1>	RG210
<2>/210	'CPI'	103 120 111	376	212<1>	IMMED
<2>/216	'DAA'	104 101 101	047	144<1>	DIRCT
<2>/224	'DCR'	104 103 122	005	245<1>	RG543
<2>/232	'DCX'	104 103 130	013	264<1>	RG54B
<2>/240	'INR'	111 116 122	004	245<1>	RG543
<2>/246	'INX'	111 116 130	003	264<1>	RG54B
<2>/254	'MOV'	115 117 126	100	145<1>	MOVRT
<2>/262	'MVI'	115 126 111	006	205<1>	MVIRT
<2>/270	'NOP'	116 117 120	000	144<1>	DIRCT
<2>/276	'ORA'	117 122 101	260	152<1>	RG210
<2>/304	'ORI'	117 122 111	366	212<1>	IMMED
<2>/312	'RAL'	122 101 114	027	144<1>	DIRCT
<2>/320	'RAR'	122 101 122	037	144<1>	DIRCT
<2>/326	'RLC'	122 114 103	007	144<1>	DIRCT
<2>/334	'RRC'	122 122 103	017	144<1>	DIRCT
<2>/342	'SBB'	123 102 102	230	152<1>	RG210
<2>/350	'SBI'	123 102 111	336	212<1>	IMMED
<2>/356	'STC'	123 124 103	067	144<1>	DIRCT
<2>/364	'SUB'	123 125 102	220	152<1>	RG210
<2>/372	'SUI'	123 125 111	326	212<1>	IMMED
<3>/000	'XRA'	130 122 101	250	152<1>	RG210
<3>/006	'XRI'	130 122 111	356	212<1>	IMMED



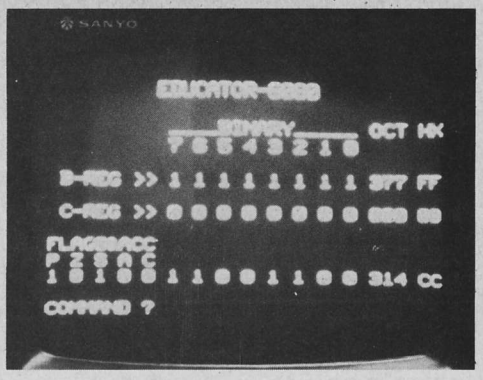


Photo 3: To illustrate the use of hexadecimal immediate values, this photo shows the operation of XRI H'C3'.

Listing 1: The Educator-8080 program expressed as an absolute assembly language listing. The notations <0>, <1>, <2> and <3> are used to denote the high order (page) address bytes of four consecutive pages in memory address space. When loading the program into a given system, these notations become bytes with consecutive octal values. Thus to load the program at location 200/000 in memory address space, the values utilized would be 200, 201, 202 and 203.

```
address      octal-code      label   op      operand     commentary

*The control routine is the top of the structure and controls the operation of the entire program.

<0>/000  061 xxx xxx  CNTRL   LXI     SP,STACK    Set stack pointer to programmable memory;
<0>/003  315 026 <0>  NOTZER  CALL    DSPLY       Display contents of registers;
<0>/006  315 316 <0>          CALL    CMDNT       Enter a command;
<0>/011  315 063 <1>          CALL    FETCH       Fetch the correct opcode;
<0>/014  267                  ORA     A           Set zero flag as per contents;
<0>/015  302 003 <0>          JNZ     NOTZER      Jump if not zero error occurred;
<0>/020  315 030 <2>          CALL    XQTER       Go execute the current command;
<0>/023  303 003 <0>          JMP     NOTZER      Loop forever;

*This display routine controls the generation of the dynamic display.

<0>/026  041 167 <3>  DSPLY   LXI     H,TITLS     Load address of titles into HL;
<0>/031  315 261 <0>          CALL    CHEDT       Display titles;
<0>/034  041 257 <3>          LXI     H,BLINE     Load addr of BLINE title;
<0>/037  072 351 <3>          LDA     BREG        Load contents of BREG into A;
<0>/042  315 132 <0>          CALL    DSPCV       Convert and display;
<0>/045  041 271 <3>          LXI     H,CLINE     Load addr of CLINE title;
<0>/050  072 350 <3>          LDA     CREG        Load contents of CREG into A;
<0>/053  315 132 <0>          CALL    DSPCV       Convert and display;
<0>/056  041 304 <3>          LXI     H,AFHDR     Load addr of A'flags title;
<0>/061  315 261 <0>          CALL    CHEDT       Display titles;
<0>/064  052 346 <3>          LHLD    PSWA        Load flags and A into HL;
<0>/067  175                  MOV     A,L         Move flags to A;
<0>/070  346 004              ANI     B'00000100' AND off all but parity flag;
<0>/072  315 237 <0>          CALL    DSPFG       Display the flag value;
<0>/075  175                  MOV     A,L         Move flags to A;
<0>/076  346 100              ANI     B'01000000' AND off all but zero flag;
<0>/100  315 237 <0>          CALL    DSPFG       Display the flag value;
<0>/103  175                  MOV     A,L         Move flags to A;
<0>/104  346 200              ANI     B'10000000' AND off all but sign flag;
<0>/106  315 237 <0>          CALL    DSPFG       Display the flag value;
<0>/111  175                  MOV     A,L         Move flags to A;
<0>/112  346 020              ANI     B'00010000' AND off all but auxiliary carry flag;
<0>/114  315 237 <0>          CALL    DSPFG       Display the flag value;
<0>/117  175                  MOV     A,L         Move flags to A;
<0>/120  346 001              ANI     B'00000001' AND off all but carry flag;
<0>/122  315 237 <0>          CALL    DSPFG       Display the flag value;
<0>/125  174                  MOV     A,H         Move A register value to A;
<0>/126  315 137 <0>          CALL    DSPCN       Display with no title print;
<0>/131  311                  RET                 Return to the CNTRL routine;

*The display conversion routine prints binary, octal and hexadecimal.

<0>/132  365          DSPCV   PUSH    PSW         Save output value for CHEDT;
<0>/133  315 261 <0>          CALL    CHEDT       Display line title addr in HL;
<0>/136  361                  POP     PSW         Retrieve saved output value;
<0>/137  036 010      DSPCN   MVI     E,Q'010'    Move 8 to E register;
<0>/141  007          DSPBT   RLC                 Rotate MSB into Carry and LSB;
<0>/142  365                  PUSH    PSW         Save current value;
<0>/143  346 001              ANI     Q'001'      AND off all but LSB;
<0>/145  315 237 <0>          CALL    DSPFG       Go display bit value;
<0>/150  361                  POP     PSW         Retrieve saved current value;
<0>/151  035                  DCR     E           Decrement loop count;
<0>/152  302 141 <0>          JNZ     DSPBT       Jump if loop count not zero;
<0>/155  267                  ORA     A           Reset carry;
<0>/156  036 003              MVI     E,Q'003'    Move 3 to E register;
<0>/160  027          DSPQT   RAL                 MSB to Carry, Carry to LSB,
<0>/161  027                  RAL                   do it again,
<0>/162  027                  RAL                     three times for octal digit shift;
<0>/163  365                  PUSH    PSW         Save current value;
<0>/164  346 007              ANI     Q'007'      AND off all but octal LSD;
<0>/166  366 060              ORI     Q'060'      OR on bits to make ASCII numeric character;
<0>/170  315 xxx xxx          CALL    CHRPR       Output the character;
<0>/173  361                  POP     PSW         Retrieve saved current value;
<0>/174  035                  DCR     E           Decrement loop count;
<0>/175  302 160 <0>          JNZ     DSPQT       Jump if loop count not zero;
<0>/200  315 251 <0>          CALL    DSPSP       Output a space;
<0>/203  036 002              MVI     E,Q'002'    Move 2 to E;
<0>/205  007          DSPHT   RLC                 Rotate MSB into Carry and LSB,
<0>/206  007                  RLC                   do it again,
<0>/207  007                  RLC                     four times for,
<0>/210  007                  RLC                       hexadecimal shift;
<0>/211  365                  PUSH    PSW         Save current value;
<0>/212  346 017              ANI     B'00001111' AND off all but hexadecimal LSD;
<0>/214  306 060              ADI     Q'060'      Add on bits to make ASCII numeric character;
<0>/216  376 072              CPI     Q'072'      Compare result to one more than 9;
<0>/220  332 225 <0>          JC      DSPHS       If numeric then skip adjustment;
<0>/223  306 007              ADI     Q'007'      Add 7 giving ASCII 'A' thru 'F' codes;
```

three consecutive numeric characters which have octal digit values of from 0 to 7.

Example: **Q'307'** V is **307**, quotes are optional.

T=H: V is a pair of consecutive characters which have hexadecimal digit values from 0 to F.

Example: **H'C7'** V is **C7**, quotes are optional.

With the exception of the move immediate (MVI) command which requires a destination register, immediate commands are entered as the mnemonic opcode followed by the immediate operand in any of its valid forms.

Some "before and after" examples of Educator-8080 commands are shown in photos 1 through 3. In each case, a command is typed into the keyboard of the computer, then the Educator-8080 display following the command is depicted.

Entering Commands

Commands are entered into Educator-8080 as a string of characters (eg: letters, numbers, spaces and punctuation) followed by a command termination character. As written, Educator-8080 assumes that the command termination character will be an ASCII carriage-return (octal 015). However, any other keyboard character code may be used as the command termination character by changing the value of the immediate operand in the instruction located at address <0>/341 which tests for command termination. (See listing 1.)

Since it is not uncommon to make errors when keying information into a computer, two provisions have been made in Educator-8080 for correcting or eliminating

Listing 1, continued:

```
    address     octal-code      label    op      operand        commentary

→   <0>/225     315 xxx xxx     DSPHS    CALL    CHRPR          Output the character;
    <0>/230     361                      POP     PSW            Retrieve saved current value;
    <0>/231     000                      NOP
    <0>/232     035                      DCR     E              Decrement loop count;
    <0>/233     302 205 <0>              JNZ     DSPHT          Jump if loop count not zero;
    <0>/236     311                      RET                    Return to calling routine;

    *Display flag or binary digit followed by a space. Alternate entry is used to display a space.

    <0>/237     312 244 <0>     DSPFG    JZ      DSPFZ          Jump if passed value is a zero;
    <0>/242     076 001                  MVI     A,Q'001'       Otherwise move a 1 into A;
    <0>/244     306 060         DSPFZ    ADI     Q'060'         Convert into ASCII numeric character;
→   <0>/246     315 xxx xxx              CALL    CHRPR          Output the character;
    <0>/251     365             DSPSP    PUSH    PSW            Save the flags and value in A;
    <0>/252     076 040                  MVI     A,Q'040'       Move space into A;
→   <0>/254     315 xxx xxx              CALL    CHRPR          Output the space;
    <0>/257     361                      POP     PSW            Retrieve the saved flags and A;
    <0>/260     311                      RET                    Return to the calling routine;

    *The character string output edit routine.

    <0>/261     176             CHEDT    MOV     A,M            Move next character into A;
    <0>/262     376 200                  CPI     Q'200'         Compare it to 200 octal;
    <0>/264     310                      RZ                     Return if equal it's end of string;
    <0>/265     322 277 <0>              JNC     CHSPA          Jump if greater for space routine;
→   <0>/270     315 xxx xxx              CALL    CHRPR          Else go output the character;
    <0>/273     043             CHEND    INX     H              Increment the string index;
    <0>/274     303 261 <0>              JMP     CHEDT          Loop for next character;
    <0>/277     326 200         CHSPA    SUI     Q'200'         Subtract 200 octal from value;
    <0>/301     107                      MOV     B,A            Move space count to B;
    <0>/302     076 040         CHSPL    MVI     A,Q'040'       Move space to A;
→   <0>/304     315 xxx xxx              CALL    CHRPR          Output the space;
    <0>/307     005                      DCR     B              Decrement space count;
    <0>/310     302 302 <0>              JNZ     CHSPL          Jump if count not zero to start of loop;
    <0>/313     303 273 <0>              JMP     CHEND          Jump back into CHEDT loop;

    *The command entry routine accepts input from the keyboard for commands.

    <0>/316     041 332 <3>     CMDNT    LXI     H,CMDMS        Move address of 'COMMAND ? ' to HL;
    <0>/321     315 261 <0>              CALL    CHEDT          Display the message;
    <0>/324     041 352 <3>              LXI     H,CMDAR        Move address of command input area HL;
    <0>/327     006 026                  MVI     B,Q'026'       Move maximum length to B;
→   <0>/331     315 xxx xxx     CMDKB    CALL    KEYBD          Get an input character;
    <0>/334     376 014                  CPI     Q'014'         Is it a control-1 line delete?
    <0>/336     312 000 <0>              JZ      CNTRL          If so then restart program;
    <0>/341     376 015                  CPI     Q'015'         Is it a carriage return?
    <0>/343     312 376 <0>              JZ      CMDND          If so then go compress input;
    <0>/346     376 177                  CPI     Q'177'         Is it a delete character?
    <0>/350     302 355 <0>              JNZ     CMDST          If not then go store the character;
    <0>/353     076 033                  MVI     A,Q'033'       If so replace with back arrow;
    <0>/355     167             CMDST    MOV     M,A            Store input character in command buffer;
→   <0>/356     315 xxx xxx              CALL    CHRPR          Display the input character;
    <0>/361     043                      INX     H              Increment command work area index;
    <0>/362     005                      DCR     B              Decrement command length count;
    <0>/363     302 331 <0>              JNZ     CMDKB          If not full then reiterate;
    <0>/366     076 001                  MVI     A,Q'001'       If buffer full then select error
    <0>/370     315 063 <2>              CALL    ERROR            number 1 and print its message;
    <0>/373     303 000 <0>              JMP     CNTRL          Restart the program;

    *The command compress routine eliminates all but letters and numbers.

    <0>/376     041 352 <3>     CMDND    LXI     H,CMDAR        Load HL with address of work area;
    <1>/001     345                      PUSH    H              Push & pop move it to DE
    <1>/002     321                      POP     D                as the compression pointer;
    <1>/003     076 026                  MVI     A,Q'026'       Load A with maximum length;
    <1>/005     220                      SUB     B              Subtract remaining length from B;
    <1>/006     107                      MOV     B,A            Move actual length to B;
    <1>/007     176             CMDNX    MOV     A,M            Move command character to A;
    <1>/010     376 033                  CPI     Q'033'         Is it a back arrow (character delete)?
    <1>/012     302 027 <1>              JNZ     CMDCH          If not then go to other tests;
    <1>/015     076 352                  MVI     A,CMDAR-L      Low address byte of CMDAR to A;
    <1>/017     273                      CMP     E              Compare to current low address byte;
    <1>/020     322 055 <1>              JNC     CMDNS          If not greater then skip save;
    <1>/023     033                      DCX     D              Else back up compression pointer;
    <1>/024     303 055 <1>              JMP     CMDNS          Skip saving the character;
    <1>/027     376 060         CMDCH    CPI     Q'060'         Is the character less than '0'?
    <1>/031     332 055 <1>              JC      CMDNS          If so then skip saving it;
    <1>/034     376 072                  CPI     Q'072'         Is the character less than '9' + 1?
    <1>/036     332 053 <1>              JC      CMDSV          If so then save numeric value;
    <1>/041     376 101                  CPI     Q'101'         Is the character less than 'A'?
    <1>/043     332 055 <1>              JC      CMDNS          If so then skip saving it;
    <1>/046     376 133                  CPI     Q'133'         Is the character greater than 'Z'?
    <1>/050     322 055 <1>              JNC     CMDNS          If so then skip saving it;
    <1>/053     022             CMDSV    STAX    D              Store character in compressed area;
    <1>/054     023                      INX     D              Increment compression pointer index;
    <1>/055     043             CMDNS    INX     H              Increment input string pointer;
    <1>/056     005                      DCR     B              Decrement actual length count;
    <1>/057     302 007 <1>              JNZ     CMDNX          If length is not zero then reiterate;
    <1>/062     311                      RET                    Else return to CNTRL calling point;

    *The FETCH instruction/command routine validates and builds the object code.

    <1>/063     041 122 <2>     FETCH    LXI     H,OPTAB        Load address of opcode table HL;
    <1>/066     036 037                  MVI     E,Q'037'       Move table element count to E;
    <1>/070     345             FLOOP    PUSH    H              Save current element address;
    <1>/071     001 352 <3>              LXI     B,CMDAR        Load address of CMDAR into BC;
    <1>/074     026 003                  MVI     D,Q'003'       Move opcode length to D;
    <1>/076     012             FCOMP    LDAX    B              Load command character to A indexed by B;
    <1>/077     276                      CMP     M              Compare it to table character;
    <1>/100     302 125 <1>              JNZ     FNXEL          If not equal then go to next element;
    <1>/103     003                      INX     B              Increment command character index;
    <1>/104     043                      INX     H              Increment table character index;
    <1>/105     025                      DCR     D              Decrement opcode length counter;
    <1>/106     302 076 <1>              JNZ     FCOMP          If not zero continue test loop;
    <1>/111     343                      XTHL                   Exchange HL with top of stack;
    <1>/112     341                      POP     H              Pop HL from stack to clear it;
    <1>/113     136                      MOV     E,M            Move naked opcode to E, D is zero;
    <1>/114     325                      PUSH    D              Save naked opcode;
    <1>/115     043                      INX     H              Increment table pointer;
    <1>/116     136                      MOV     E,M            Decode routine low address byte to E;
    <1>/117     043                      INX     H              Increment table pointer;
    <1>/120     126                      MOV     D,M            Decode routine high address byte to D;
    <1>/121     353                      XCHG                   Move decode routine address to HL;
```

errors. The ASCII delete character code (octal 177) is used to delete the last remaining character in the input string. Since a deleted character is not considered to exist, N consecutive delete characters will delete the N preceding characters. For example, if the delete character is shown as a back arrow (←), RAX←L will be reduced to RAL and CQP←←MA will be reduced to CMA. Characters which have been keyed in are displayed after they have been tested. The display function uses the octal value 177 as a clear screen control code; therefore, character deletes are transformed into the back arrow before they are displayed and stored. Educator-8080 users with systems which have a back arrow (octal 033) key on their keyboards may use it as a character delete code and it will have the same effect as the delete key assumed in this version. Users who have neither of these keys can designate any keyboard character as the delete character code by changing the immediate operand in the instruction located at <0>/346 which tests for the delete character. (See listing 1). The other, and somewhat more drastic, method of eliminating keying errors is to delete the entire input line. This is usually done when an error is detected before the command termination character is input but several characters after the error occurred. The procedure for deleting an entire line is to enter an ASCII form feed code (octal 014) which is a "control L" combination on typical ASCII keyboards. This will clear the input line and restart the command entry procedure. Like the command termination and the character delete codes, the line delete code can be made to be any keyboard character by changing the value of the immediate operand in the instruction at location <0>/334 which tests for the line delete code.

A very useful feature of Educator-8080 permits the user to execute the last command input several times. This is accomplished by simply keying the command termination character when the system calls for the entry of a new command. In order to provide this facility the input buffer is not cleared prior to calling for the entry of a new command, so the last previously entered command is still in the buffer. This feature is especially handy when demonstrating the effect of multiple executions of the rotate, increment, decrement, arithmetic and logical commands.

The general format for entering a command is as follows:

OPCODE[p OPERAND-1[p OPERAND-2]]t

Where:

OPCODE is the mnemonic opcode for the

command. For example; MOV XRI, etc.

p — is any desired form of punctuation or a space. p is not required and, therefore, may be omitted entirely.

OPERAND-1 — is the first or only operand required by an instruction. It may be a register identification or an immediate value. See table 1 for the operand requirements.

OPERAND-2 — is the second operand where required by a specific instruction. See table 1.

t — is the command termination character, an ASCII carriage return in the listing 1 version of Educator-8080.

The brackets ([ ]) shown in the general format are used to indicate that the items within them are optional, since some commands do not require any operands (eg: RAL, STC, CMA, etc), some require one operand only (eg: ADI, CMP, XRA, etc), and some commands require two operands (eg: MOV and MVI).

Error Messages

In the process of entering and executing commands under Educator-8080 there are a number of errors which can occur. When this happens an error message is displayed on the output device. For the benefit of users with television displays, a delay of approximately two seconds occurs as the message is being displayed, to provide time to read it. After the two second delay the normal Educator-8080 display is generated and the command entry mode is reentered. Teletype or other hard copy users will probably wish to alter the error display routine slightly by eliminating the extraneous spaces which are used to center the error messages on the TV monitor screen.

The errors which can occur are listed in absolute octal form in table 3. The error numbers and extended explanations of conditions are as follows:

1. **INPUT TOO LONG:** The input string exceeds 22 characters in length probably because too many characters were deleted since delete character codes count as input characters. Twenty two characters should be sufficient for any normal entry including punctuation and several character deletes.
2. **INVALID COMMAND:** The input command mnemonic is not one of the ones implemented by Educator-8080.
3. **INVALID REGISTER:** The operand register is not A, B or C for a command which requires a single register as an operand or it was not B

Listing 1, continued:

```
address      octal-code        label   op      operand     commentary
<1>/122  321                           POP     D           Unsave naked opcode to DE;
<1>/123  257                           XRA     A           Clear A, no error code;
<1>/124  351                           PCHL                Jump to address of decode routine;
<1>/125  001  006  000         FNXEL   LXI     B,Q'000006' Load double length 6 into BC;
<1>/130  341                           POP     H           Unsave current element address;
<1>/131  011                           DAD     B           Add 6 to it;
<1>/132  035                           DCR     E           Decrement table element count;
<1>/133  302  070  <1>                 JNZ     FLOOP       Reiterate to test next element;
<1>/136  076  002                      MVI     A,Q'002'    Move error code 2 to A;
<1>/140  303  063  <2>                 JMP     ERROR       Go display error 2, opcode unknown;
<1>/143  000                           NOP                 No operation filler;

*The instruction decoder routines follow.

*Instructions using the DIRCT routine require no decoding. Example RAL, CMA, etc.

<1>/144  311                   DIRCT   RET                 Return to CNTRL for execution;

*The MOVRT is used only by the MOV command.

<1>/145  315  245  <1>  MOVRT  CALL    RG543       Validate destination register;
<1>/150  267                   ORA     A           Set flags based on A contents;
<1>/151  300                   RNZ                 Return not zero with error;
                                                   Else fall thru to RG210;

*Instructions using the RG210 routine require a source register.

<1>/152  012                   RG210   LDAX    B   Load next command character into A;
<1>/153  003                           INX     B   Increment command character index;
<1>/154  315  173  <1>                 CALL    REGAN   Analyze for valid register;
<1>/157  322  166  <1>                 JNC     RGERR   If CY=0 then register not valid;
<1>/162  203                           ADD     E   Add naked opcode to register value;
<1>/163  137                           MOV     E,A Move result back to E;
<1>/164  257                           XRA     A   Clear A indicating no errors;
<1>/165  311                           RET         Return to CNTRL;

*The register error routine is used to indicate register designation errors.

<1>/166  076  003              RGERR   MVI     A,Q'003'    Move error code 3 to A;
<1>/170  303  063  <2>                 JMP     ERROR       Go display error 3, invalid register;

*The register analysis and validation routine is used by RG543, RG210 and RG54B.

<1>/173  326  101              REGAN   SUI     Q'101'      Subtract an 'A' from the character;
<1>/175  376  003                      CPI     Q'003'      Compare the result to 3;
<1>/177  320                           RNC                 If not less than 3 return with CY=0;
<1>/200  075                           DCR     A           Decrement result: A=377, B=000, C=001;
<1>/201  346  007                      ANI     Q'007'      AND off all but octal LSD;
<1>/203  067                           STC                 Set CY=1 indicating no error;
<1>/204  311                           RET                 Return to calling routine;

*The MVIRT is used only by the MVI command.

<1>/205  315  245  <1>  MVIRT  CALL    RG543       Validate destination register;
<1>/210  267                   ORA     A           Set flags based on A contents;
<1>/211  300                   RNZ                 Return not zero with error;
                                                   Else fall thru to IMMED;

*Instructions requiring an immediate operand use the IMMED routine.

<1>/212  012                   IMMED   LDAX    B   Load next command character into A;
<1>/213  003                           INX     B   Increment command character index;
<1>/214  376  102                      CPI     Q'102'  Is the command character a 'B'?
<1>/216  312  301  <1>                 JZ      BINRY   If so then process as binary;
<1>/221  376  121                      CPI     Q'121'  Is the command character a 'Q'?
<1>/223  312  336  <1>                 JZ      OCTAL   If so then process as octal;
<1>/226  376  110                      CPI     Q'110'  Is the command character an 'H'?
<1>/230  312  367  <1>                 JZ      HEX     If so then process as hexadecimal;
<1>/233  376  070                      CPI     Q'070'  Is the command character less than '8'?
<1>/235  332  335  <1>                 JC      OCTAD   If so then treat as octal;
<1>/240  076  005                      MVI     A,Q'005'    Move error code 5 to A;
<1>/242  303  063  <2>                 JMP     ERROR       Go display error 5, invalid immediate;

*Instructions using the RG543 routine require a destination register.

<1>/245  012                   RG543   LDAX    B   Load next command character into A;
<1>/246  003                           INX     B   Increment command character index;
<1>/247  315  173  <1>                 CALL    REGAN   Analyze for valid register;
<1>/252  322  166  <1>                 JNC     RGERR   If CY=0 then register not valid;
<1>/255  007                           RLC             Shift octal register value
<1>/256  007                           RLC               left three
<1>/257  007                           RLC                 places;
<1>/260  203                           ADD     E   Add naked opcode to shifted value;
<1>/261  137                           MOV     E,A Move result back to E;
<1>/262  257                           XRA     A   Clear A indicating no errors;
<1>/263  311                           RET         Return to calling routine;

*Instructions using the RG54B routine are INX and DCX.

<1>/264  012                   RG54B   LDAX    B   Load next command character into A;
<1>/265  003                           INX     B   Increment command character index;
<1>/266  315  173  <1>                 CALL    REGAN   Analyze for valid register;
<1>/271  376  000                      CPI     Q'000'  Is the register a zero?
<1>/273  310                           RZ              If so it's 'B' so return;
<1>/274  076  004                      MVI     A,Q'004'    Move error code 4 to A;
<1>/276  303  063  <2>                 JMP     ERROR       Go display error 4, invalid register;

*The BINRY routine converts a binary immediate value into usable form.

<1>/301  046  010              BINRY   MVI     H,Q'010'    Move 8 to H for count;
<1>/303  012                   BLOOP   LDAX    B   Load next command character into A;
<1>/304  326  060                      SUI     Q'060'  Subtract a '0' from it;
<1>/306  376  002                      CPI     Q'002'  Is the result less than 2?
<1>/310  322  330  <1>                 JNC     IMMER   If not then go display immediate error;
<1>/313  345                           PUSH    H       Save the count;
<1>/314  152                           MOV     L,D     Move D to L (immediate byte);
<1>/315  051                           DAD     H       Shift HL left one bit;
<1>/316  205                           ADD     L       Add L to bit in A;
```

Listing 1, continued:

```
address      octal-code      label   op      operand     commentary
<1>/317      127             MOV     D,A     Move the result back to D;
<1>/320      341             POP     H       Unsave the count;
<1>/321      003             INX     B       Increment command character index;
<1>/322      045             DCR     H       Decrement the count;
<1>/323      302 303 <1>     JNZ     BLOOP   If not zero then reiterate;
<1>/326      257             XRA     A       Clear A indicating no errors;
<1>/327      311             RET             Return to CNTRL;

*The immediate error routine is used to indicate immediate value errors.

<1>/330      076 006         IMMER   MVI     A,Q'006'    Move error code 3 to A;
<1>/332      303 063 <2>             JMP     ERROR       Go display error 3, invalid immediate;

*The OCTAD entry point to the OCTAL routine is for the default condition.

<1>/335      013             OCTAD   DCX     B           Decrement command character index;

*The OCTAL routine converts an octal immediate value into usable form.

<1>/336      046 003         OCTAL   MVI     H,Q'003'    Move a 3 into H for count;
<1>/340      012             OLOOP   LDAX    B           Load next command character into A;
<1>/341      326 060                 SUI     Q'060'      Subtract a '0' from it;
<1>/343      376 010                 CPI     Q'010'      Is command character less than 8?
<1>/345      322 330 <1>             JNC     IMMER       If not then go display immediate error;
<1>/350      345                     PUSH    H           Save the count;
<1>/351      152                     MOV     L,D         Move D to L immediate byte;
<1>/352      051                     DAD     H           Shift immediate
<1>/353      051                     DAD     H             byte left
<1>/354      051                     DAD     H               three bits;
<1>/355      205                     ADD     L           Add L to value in A;
<1>/356      127                     MOV     D,A         Move result back to D;
<1>/357      341                     POP     H           Unsave the count;
<1>/360      003                     INX     B           Increment command character index;
<1>/361      045                     DCR     H           Decrement the count;
<1>/362      302 340 <1>             JNZ     OLOOP       If not zero then reiterate;
<1>/365      257                     XRA     A           Clear A indicating no errors;
<1>/366      311                     RET                 Return to CNTRL;

*The HEX routine converts a hexadecimal immediate value into usable form.

<1>/367      046 002         HEX     MVI     H,Q'002'    Move a 2 into H for count;
<1>/371      012             HLOOP   LDAX    B           Load next command character into A;
<1>/372      326 060                 SUI     Q'060'      Subtract a '0' from it;
<1>/374      376 012                 CPI     Q'012'      Is it less than '9' + 1?
<1>/376      332 010 <2>             JC      HCHOK       If so then numeric character is OK;
<2>/001      326 007                 SUI     Q'007'      Else convert alphabetic to numeric;
<2>/003      376 020                 CPI     Q'020'      Is character value greater than 15?
<2>/005      322 330 <1>             JNC     IMMER       If so then invalid hexadecimal value;
<2>/010      345             HCHOK   PUSH    H           Save the count;
<2>/011      152                     MOV     L,D         Move D to L immediate byte;
<2>/012      051                     DAD     H           Shift immediate
<2>/013      051                     DAD     H             byte left
<2>/014      051                     DAD     H               four
<2>/015      051                     DAD     H                 bits;
<2>/016      205                     ADD     L           Add L to value in A;
<2>/017      127                     MOV     D,A         Move result back to D;
<2>/020      341                     POP     H           Unsave the count;
<2>/021      003                     INX     B           Increment command character index;
<2>/022      045                     DCR     H           Decrement the count;
<2>/023      302 371 <1>             JNZ     HLOOP       If not zero then reiterate;
<2>/026      257                     XRA     A           Clear A indicating no errors;
<2>/027      311                     RET                 Return to CNTRL;

*The XQTER routine executes the generated object code for Educator-8080.

<2>/030      353             XQTER   XCHG                Move generated opcode to HL;
<2>/031      042 046 <2>             SHLD    XQTOP       Store it at execution point;
<2>/034      052 346 <3>             LHLD    PSWA        Load working PSW & A into HL;
<2>/037      345                     PUSH    H           Push & pop sets values for
<2>/040      361                     POP     PSW           working register and flags;
<2>/041      052 350 <3>             LHLD    BANDC       Load working B and C into HL;
<2>/044      345                     PUSH    H           Push & pop sets values for
<2>/045      301                     POP     B             working B and C registers;
<2>/046      000             XQTOP   NOP                 The command to be executed;
<2>/047      000                     NOP                 Immediate value or NOP;
<2>/050      305                     PUSH    B           Push B and C working register values;
<2>/051      341                     POP     H           Pop them into HL;
<2>/052      042 350 <3>             SHLD    BANDC       Store them in save area;
<2>/055      365                     PUSH    PSW         Push PSW and A working values;
<2>/056      341                     POP     H           Pop them into HL;
<2>/057      042 346 <3>             SHLD    PSWA        Store them in save area;
<2>/062      311                     RET                 Return to CNTRL for next command;

*The ERROR routine is used to display error messages.

<2>/063      365             ERROR   PUSH    PSW         Save error code in A;
<2>/064      041 162 <3>             LXI     H,ERRSP     Load address of error header spaces;
<2>/067      315 261 <0>             CALL    CHEDT       Go output error header spaces
<2>/072      361                     POP     PSW         Unsave error code;
<2>/073      041 014 <3>             LXI     H,ERTAB     Load address of error message table;
<2>/076      205                     ADD     L           Add low address byte to error code;
<2>/077      157                     MOV     L,A         Move result to L, points to offset;
<2>/100      156                     MOV     L,M         Move offset to L;

*Note: HL now contains the address of the error message.

<2>/101      315 261 <0>             CALL    CHEDT       Output the error message;
<2>/104      021 000 000     ERTIM   LXI     D,Q'000000' Load DE with timing loop value;
<2>/107      035                     DCR     E           Decrement value in E 256 times;
<2>/110      302 105 <2>             JNZ     ERTIM+1     Reiterate loop 256 times;

*The above JMP goes to the first 000 in the LXI command which is an effective NOP.

<2>/113      025                     DCR     D           Decrement D;
<2>/114      302 105 <2>             JNZ     ERTIM+1     Reiterate outer loop 256 times;
<2>/117      076 377                 MVI     A,Q'377'    Move a 377 to A indicating error;
<2>/121      311                     RET                 Return to CNTRL;

*Note: for Teletype or hard copy output bytes <2>/104 thru <2>/116 can be replaced by 000 NOPs.
```

for the INX or DCX commands which require register pairs as operands.

4. **INVALID IMMED TYPE:** The type code for an immediate operand is not B, Q, or H, or if the default was attempted the first digit of the implied octal value was not a digit from 0 to 7.
5. **INVALID IMMED VALUE:** One of the characters in the immediate operand value string was inconsistent with the immediate type code. For example, a digit in a binary input string was not a zero or a one. This can also be caused by not providing the correct quantity of digits for the immediate type specified; too few digits will possibly cause a problem. If too many digits are entered only the first N will be used (N=8 for binary, N=3 for octal and N=2 for hexadecimal).
6. **ERROR!** This message should not occur unless a grave internal error occurs in Educator-8080.

Educator-8080 Program Listing

The Educator-8080 program is presented in an assembly language format as listing 1. It was hand assembled and, therefore, some liberties were taken in the way it was presented. Addresses are shown in a split octal ("Intelese") format of page and address within page. Educator-8080 requires four contiguous 256 byte pages of memory (it just fits); to ease the implementation process all addresses and address sensitive bytes are shown with relative page numbers in the format <P>, where P is a 0, 1, 2 or 3. A simple process of substitution as the program is being put into the machine will provide the ability to locate Educator-8080 in any four contiguous pages provided the program begins on a page boundary.

The assignment of three addresses is left to the user. These three addresses are shown symbolically in both the source and the object code. The first address is for the location of the STACK; insert the address of the stack in the command at location <0>/000. The stack should be capable of being at least 10 to 12 levels deep to function correctly. The second and third addresses are the addresses of the physical input and output routines which must be provided by the user. These routine addresses are shown symbolically as KEYBD and CHRPR in the source listing. The values are shown as 'XXX XXX' in the object code.

Input and Output Routines

The Educator-8080 program references two subroutines for the purpose of exe-

Table 3:
Error Messages. This table consists of a list of address offsets (location <3>/014) followed by the ASCII error message strings. The octal values 201 through 377 are used to encode from 1 to 177 spaces (1 to 127 decimal). The strings contain a single space for these codes. The octal value 200 is used to indicate end of string, and is shown symbolically as the character "▽". The octal value 177 is used to indicate the clear screen operation, and is shown symbolically as the character "■".

cuting IO operations. The KEYBD subroutine is used to read a single character of input from an ASCII keyboard device. The CHRPR subroutine is used to display (or print) a single character. These routines are not shown in the listings, but should be adapted from the routines normally used with the particular system in which the program is run. Both KEYBD and CHRPR use the accumulator (A) to pass a single character argument. KEYBD defines a value in A obtained from the input device. CHRPR displays the value in A on a device such as a video display or Teletype. All other registers of the 8080 processor should be left unchanged upon return from either of these routines. Entry to the IO routines is shown using a CALL instruction in these listings. A corresponding RET instruction in the routine should return control when either operation is completed. An alternate method of entry would be to employ the 8080 RST instruction in place of CALL. If the Educator-8080 listings accompanying this article are used without reassembly, then the CALL instructions would be replaced by an RST and two single byte NOP instructions.

The keyboard entry routine KEYBD

Continued on page 75

Address	Octal Code	ASCII String Value
<3>/014	153 024 043 063 063 104 127 153	Address Offsets for messages 0 through 7
<3>/024 <3>/034	111 116 120 125 124 040 124 117 117 040 114 117 116 107 200	'INPUT TOO LONG ▽ '
<3>/043 <3>/053	111 116 126 101 114 111 104 040 103 117 115 115 101 116 104 200	'INVALID COMMAND▽ '
<3>/063 <3>/073 <3>/103	111 116 126 101 114 111 104 040 122 105 107 111 123 124 105 122 200	'INVALID REGISTER▽ '
<3>/104 <3>/114 <3>/124	111 116 126 101 114 111 104 040 111 115 115 105 104 040 124 131 120 105 200	'INVALID IMMED TYPE▽ '
<3>/127 <3>/137 <3>/147	111 116 126 101 114 111 104 040 111 115 115 105 104 040 126 101 114 125 105 200	'INVALID IMMED VALUE▽ '
<3>/153	105 122 122 117 122 041 200	'ERROR!▽ '

The following string is given the name "ERRSP" and is used to clear the screen, then space down to the center prior to displaying an error message.

<3>/162	177 377 377 211 200	'■ ▽ '

Address	Octal Code	Name	ASCII Value
<3>/167	177 211 105 104 125 103 101 124	TITLS	'■ EDUCAT '
<3>/177	120 122 055 070 060 070 060 264		'OR-8080 '
<3>/207	137 137 137 137 102 111 116 101		'←←←← BINA'
<3>/217	122 131 137 137 137 137 137 040		'RY←←←← '
<3>/227	117 103 124 040 110 130 212 067		'OCT HX 7'
<3>/237	040 066 040 065 040 064 040 063		' 6 5 4 3 '
<3>/247	040 062 040 061 040 060 250 200		' 2 1 0 ▽ '
<3>/257 <3>/267	102 055 122 105 107 040 076 076 040 200	BLINE	'B-REG >> ' ' ▽ '
<3>/271 <3>/301	241 103 055 122 105 107 040 076 076 040 200	CLINE	' C-REG ' '> ▽ '
<3>/304 <3>/314 <3>/324	240 106 114 101 107 123 046 101 103 103 227 120 040 132 040 123 040 101 040 103 227 200	AFHDR	' FLAGS&A ' 'CC P Z S' ' A C ▽ '
<3>/332 <3>/342	240 103 117 115 115 101 116 104 040 077 040 200	CMDMS	'COMMAND ' '? ▽ '

Table 4: Educator-8080 standard display format messages. This table contains the definitions of several character string messages which are used to format the output display device. As in table 3, the codes from octal 201 to 377 represent from 1 to 177 spaces transmitted. The character "▽" is used to indicate an end of text code, octal 200. The character "■" is used to indicate a clear screen code, octal 177.