THE MST-80B MICROCOMPUTER TRAINER

AN INTRODUCTION TO THE MST-80

The LLL MST-80 is a complete, self-contained, microcomputer system housed in a briefcase for portability and convenience of use. It was designed at the Lawrence Livermore Laboratory.

The trainer is designed to allow students to explore and learn the hardware and software capability of the 8080 microprocessor. It includes a breadboard socket so that experiments can be interfaced to the trainer. This allows the student to learn interfacing techniques as well as programming.

A keyboard and numerical display are provided for the student to communicate with the trainer. This Input/Output (I/O) combination eliminates the need for expensive and bulky I/O such as a teletypewriter, but limits the keyboard and numerical display to communication in either the octal(base 8) number system or the hexadecimal (base 16) number system. The user can select which number system he prefers by simply depressing a control key.

A block diagram of the trainer is shown in Figure 1 and Figure 2 shows the schematic diagram of the trainer.

HARDWARE FEATURES OF THE TRAINER

1. Design is based on the 8080A CPU and support chips. The 8080A is a second-generation microcomputer CPU, with an 8-bit word and 78 instructions.

2. Has 512 bytes of read/write memory (RWM).

3. Has sockets for three 1702A PROM's (768 bytes). Also includes one uncommitted socket that can be jumper-wired to a 24 PIN ROM of user's choice. Normally a monitor program resides in PROM Ø and PROM 1.

4. Has a memory-mapped keyboard. (See Figure 3 for the memory map.)

5. Has a three digit display with full hex number capability. Ports 0,5,6,7

6. Has one 8-bit input port. Address = 1.

7. Has one 8-bit output port (latched). Address = 1.

8. Has single machine cycle step capability.

9. Has ten uncommitted LED's that can easily be connected to any desired signals (address lines, data lines, status, etc.). These can be used in single step mode.

10. Has 60 Hz timing source.

11. Has single interrupt vector capability.

Figure 4 shows the connectors used to interface the trainer and also gives detailed information on each signal and its connector pin number. Figure 5 shows the hidden connections in the breadboard mounted on the power supply. Breadboarded circuitry should not draw more than 600 MA at 5V from the computer board.

MONITOR PROGRAM

The trainer contains a monitor program that allows a user to enter a program in RWM, examine locations, change contents of locations and run the user program from a specified starting address.

The monitor program also contains a debug routine to assist the user in program debug. This routine allows the user to insert breakpoints (F7) in his program. When a breakpoint is encountered the break routine (in the monitor program) will be entered which will save all the CPU registers and the breakpoint address, and will display BB to signal the user that a breakpoint has been encountered.

The contents of the CPU registers and breakpoint address are saved in dedicated page 7 memory locations shown in Figure 3.

These locations can be examined using the DISP feature of the monitor program and, if desired, can be changed to new values using the ENTER feature of the monitor program. A detailed description of how to do this is included in the SAMPLE PROGRAM discussion.

The RUN feature of the monitor program starts the user's program with the CPU registers initialized to the current values found in these dedicated memory locations. These values may be changed before pushing RUN. A complete listing and flow chart of this program is included in the Appendix.

OPERATION OF KEYBOARD USING THE MONITOR

KEYBOARD LAYOUT

C	D	E	F	RESET	EXA
8	9	A	B	RUN	LDH
4	5	6	7	DISP	H/O
0	1	2	3	ENTER	SS

RESET: This key resets the system and starts the the monitor program running at location Ø.

NUMBER KEYS: Pushing these keys causes a number to be entered into the display in a left shift mode. Care must be exercised when entering numbers to ensure that the intended number is entered, since the display is not cleared but simply shifted left. For instance if you want to enter a 1 into the display, you should push 01 to insure the old number is completely replaced.

The current value in the display is also stored in a memory location called KYTEM.

Keys 8 through F are ignored when in OCTAL mode and are functional in HEX mode.

LDH: Load High-Order Address. In order to address any location in memory the user needs to specify the complete address. The high-order address is specified by keying the desired value into the display and then pushing LDH (LOAD H). This stores the high value in a memory location called HVALU for later use by the monitor program.

The low order address is specified by the current contents of the display whenever it is needed, i.e., in RUN or DISP operations. Its current value is kept in a memory location called LVALU.

DISP: Display. When it is desired to examine the contents of a memory location the DISP key is used. The high order address is selected by entering the desired value and using the LDH key, as explained above. The low order address is then keyed into the display, then the DISP key is pushed. This will cause the contents of the desired address to be displayed.

ENTER: The ENTER key is used to enter new values into specified locations. ENTER also automatically increments the address value, allowing the user to quickly examine or enter new values into consecutive locations in memory.

The address is set by using the DISP key since the present value should be displayed before you enter a new value. After pushing DISP a new value may be keyed into the display and when ENTER is pushed this value is entered into the currently addressed location.

In addition, the address is incremented and the contents of the next consecutive location is displayed. That value can be re-entered by pressing ENTER again or a new value can be keyed in before pressing ENTER.

RUN: This allows you to start a user program at any specified address. The address is specified by using the LDH key and keying into the display the low order address beofre pushing RUN. Remember RUN initializes all CPU registers from dedicated memory locations before starting the user program.

EXA: Examine address. Pushing this key displays the current value of the low order address. This is useful when you are examining a program (stepping through using ENTER) and you forget where you are.

H/O: Hex/Octal. This key is used to select the desired keyboard mode. When RESET is pushed after first turning on power, the keyboard will be in HEX mode. Depressing the H/O key will then cause a switch to OCTAL mode. Depressing the H/O key again will cause the mode to switch back to HEX. In short, depressing the H/O key changes the keyboard mode from the mode the system is presently in to the other mode.

SS: Single Step. This key is used in single step mode to advance the program to the next machine cycle. The toggle switch labeled SS-RUN must be in the SS position before the SS key is functional.

SAMPLE PROGRAM

Here is a sample program for the MST-80B:

MEMORY LOCATION	MACHINE CODE	OPERATIONS		
ØØ	3E		MVI A, Ø	; CLEAR AC
Ø1	ØØ			
Ø2	57	AGAIN:	MOV D, A	; SAVE A
Ø3	CD		CALL DISPLAY	; SEND AC TO DISPLAY
Ø4	52			
Ø5	Ø1			
Ø6	7A		MOV A, D	; RESTORE A
Ø7	Ø6		MVI B, Ø	; CLR B REGISTER
Ø8	ØØ			
Ø9	ØE		MVI C, 4Ø	; PUT 64 IN C REGISTER
ØA	4Ø			
ØB	Ø4	LOOP:	INR B	; INCREMENT B
ØC	CA		JZ LOOP	; DO IT AGAIN
ØD	ØB			
ØE	Ø6			
ØF	ØD		DCR C	; DECREMENT C
1Ø	C2		JNZ LOOP	; LOOP UNTIL ZERO
11	ØB			
12	Ø6			
13	C6		ADI ØØ1	; ADD ONE TO AC
14	Ø1			
15	C3		JMP AGAIN	; GO DISPLAY AC & DO AGAIN
16	Ø2			
17	Ø6			

This program can be used to demonstrate the use of the monitor program in HEX mode. Load the sample program into memory as follows:

Before you start, you need to decide where to load it. Let's put it in memory page 6 starting at location Ø (absolute address = Ø6ØØ hex). First, key Ø6 into the display and then push the LDH (load H) key. This sets the high order address (High byte) to page 6. Next key ØØ into the display, and push the DISP key. This will display the current contents of location Ø on page 6. Now you can key in the machine language code for the first instruction, 3E (MVI A), and push the ENTER key. This will enter the 3E into location Ø and will also display the contents of the next location (loc 1). Now you can key in the next code, ØØ, and push ENTER again. The ØØ will be entered into location 1 and then location 2 will be displayed. Continue this process until the entire program is entered.

If you make a mistake while keying in a number, just continue to key in until the correct value appears in the display. (The displayed number is not used until a control key is pressed.) If at any time while loading a program you forget where you are, just press EXA (examine address) and the current low order address will appear in the display. You can continue on from that point by pushing the DISP key and then the ENTER key. Or you can key in a new address into the display; then pushing the DISP key will allow you to continue from that address.

After the entire program has been keyed in, you may want to check it for correctness. This is done by keying the starting address into the display (ØØ for our sample program), pushing the DISP key and then repeatedly pushing the ENTER key. This will step through the program sequentially and display each location so it can be checked. If a mistake is found, just key in the correct value before the ENTER key is pushed.

After the program is loaded satisfactorily you can run it if so desired. To run the program, key the starting address (ØØ for our sample program) into the display and push RUN. If you are not sure what the current high order address (HVALU) is, you should set it to the correct value using the LDH key as explained previously.

USING BREAKPOINTS IN PROGRAM DEBUGGING

The use of a breakpoint in program debugging can be demonstrated using the sample program shown in Figure 6.

The program is a simple count routine that will cause the display to count up at a fixed rate determined by the constants in the counting loops. If you execute the program as it is written, you will notice the display is counting very rapidly. This is not intentional and is caused by a program bug. Let's use the breakpoint to find it.

Looking at the flow chart you can see there are two counting loops. The first one counts up to 256 and then goes back to Ø. Then the second count loop is entered. It counts the number of times the first loop must go through a full count (256 counts). Since the C register is initialized to 64, the second loop counts 64 counts, hence the total counts for both loops is 64 X 256 (= 16384) counts. After the full count is reached, 1 is added to the A register and its contents are displayed. Then the count loop starts over. This program runs endlessly until stopped by the user.

The first thing to check is to see if the registers are initialized correctly. This is done by inserting a breakpoint (breakpoint code = F7) in place of the INR B instruction at memory location Ø6ØB. (Remember to set the high order address to page 6.) Run the program. It will break when the F7 is encountered and a BB will appear in the display to signal the user that a break has occured. The break routine automatically sets HVALU to page 7 and BB is being displayed so if you now push the DISP key, the contents of memory location BB page 7 will be displayed. This location contains the low byte of the address where the break occured. The high byte of the break address is stored in location BC, so pushing the ENTER key will cause it to be displayed. Repeated use of the ENTER key allows you to examine the contents of all the CPU registers. The BREAK routine stores these away in the following memory locations:

BREAK ROUTINE MEMORY STORAGE LOCATIONS (MEMORY PAGE 7)

LOC	CONTENTS		LOC	CONTENTS
BB	PCL	Break Point	C0	B REG
BC	PCH	Address	C1	E REG
BD	PSW		C2	D REG
BE	A REG		C3	L REG
BF	C REG		C4	H REG

Register C is stored in location BF and upon examination should contain 40. Location BE (A REG) and C0 (B REG) should contain zero. If these are O.K. replace the INR B instruction (04) in location OB and put a breakpoint (F7) in location OF in place of the DCR C instruction. Run the program. When it breaks, examine location C0 again to see what the B REG is now. It should be a zero when the count loop is exited. But it is not zero! The bug must be in this loop. Upon inspection of the program it is apparent that the JZ Loop instruction, which tests for completion of the count, it testing the wrong condition. It exits the loop on nonzero count rather than zero count, so you need to replace the JZ instruction with a JNZ (C2) instruction. Replace the breakpoint in OF with DCR C (OD) and run the program. It should run O.K. with the display counting much slower.

This may appear to be trivial bug and should be apparent by just inspecting the program listing. But this is one of the most common programming errors (that is, using the wrong sense of a test instruction), and is usually quite difficult to find in a more complex program.

READ-WRITE MEMORY TEST

Included in the Monitor program is the capability of testing the resident read-write memory. Execution of this program from location 0182 will write every possible 8-bit combination of bits into the read-write memory on pages 6 and 7. The current bit pattern used for testing is displayed on the LED display. Should the written pattern not equal the read pattern, execution will halt, else it will continue cycling between pages 6 and 7.

ASSEMBLY LANGUAGE PROGRAMMING

Assembly language programming for the 8080 can be relatively easily done using table assembly for its 244 instruction-operand combinations. To help the table assembly process, three types of instruction orderings are given in the appendix.

Table 1 shows the instructions ordered as to instruction type. This table also shows instruction length in bytes, instruction execution time in clock cycles, and resulting condition flag changes if any.

Table 2 is an alphabetic listing of all 244 instruction-operand possibilities and is used for table assembly of machine code from 8080 instruction mnemonics.

Table 3 is a numerical listing of all 244 instruction-operand possibilities and can be used for disassembly of machine code.

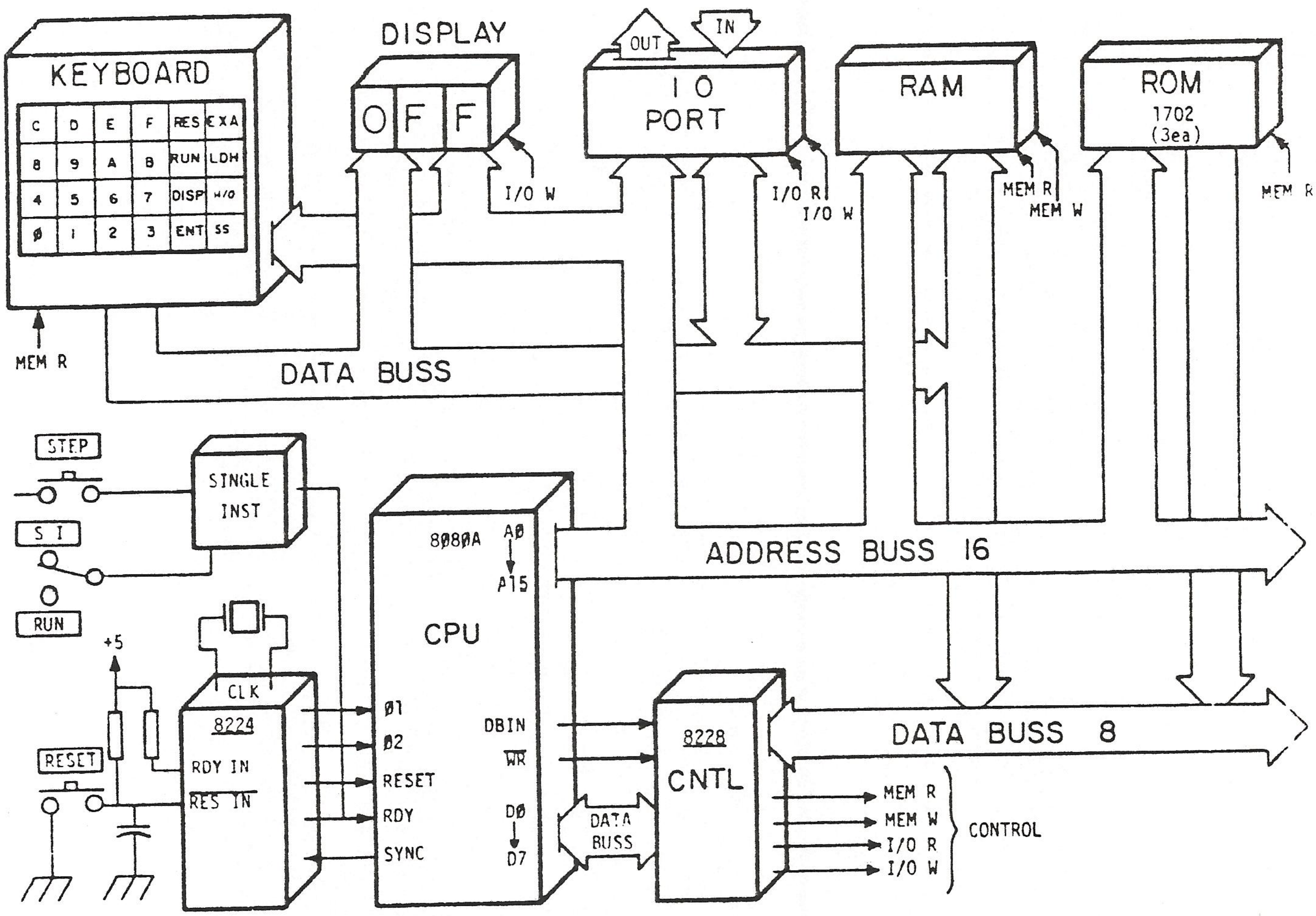
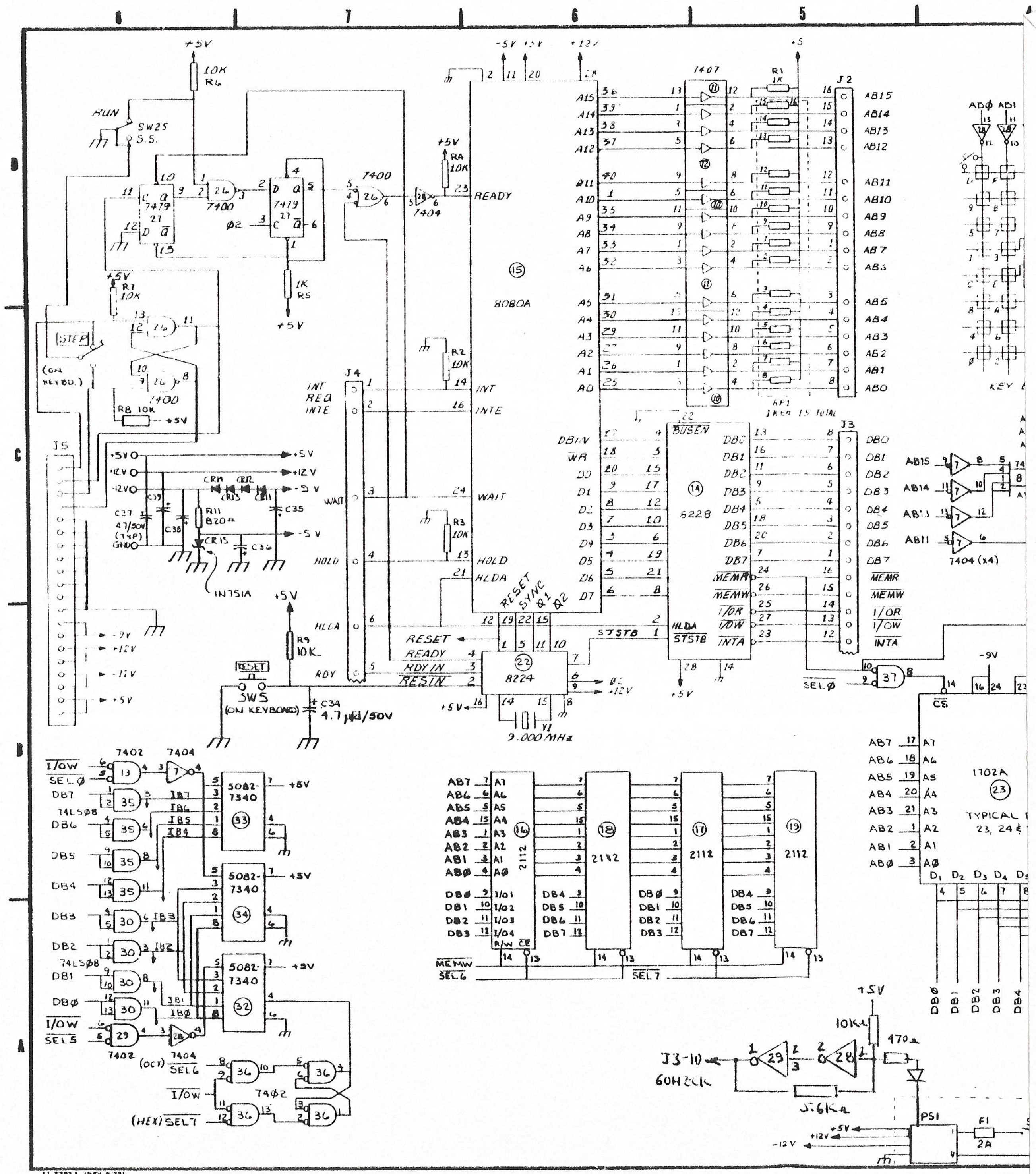


Figure 1. Operational block diagram of MST-80B Microcomputer trainer.



8080A
8228
8224
2112
2142
1702A
TYPICAL
5082-7340
7400
7404
7402
74LS08
7479
1407
J2
J3
J4
J5
RESET
READY
RDYIN
RESIN
9.000 MHz
RUN
SW25 S.S.
STEP (ON KEYBD.)
SW5 (ON KEYBOARD)
INT
REQ
INTE
WAIT
HOLD
HLDA
RDY
IN751A
J3-10 60HzCLK
7404 (x4)
MEMR
MEMW
I/OR
I/OW
INTA
SEL0
SEL5
SEL6
SEL7
(OCT) SEL6
(HEX) SEL7
LL 2703 1 (REV 8/78)

HEX		OCTAL
ØØØØ – ØØFF	PAGE Ø (PROM) MONITOR PROGRAM	ØØØØØØ – ØØØ377
Ø1ØØ – Ø1FF	PAGE 1 (PROM) MONITOR PROGRAM	ØØØ4ØØ – ØØØ777
Ø2ØØ – Ø2FF	PAGE 2 (PROM)	ØØ1ØØØ – ØØ1377
Ø3ØØ – Ø3FF	PAGE 3	ØØ14ØØ – ØØ1777
Ø4ØØ – Ø4FF	PAGE 4	ØØ2ØØØ – ØØ2377
Ø5ØØ – Ø5FF	PAGE 5 KEYBOARD	ØØ24ØØ – ØØ2777
Ø6ØØ – Ø6FF	PAGE 6 (RAM)	ØØ3ØØØ – ØØ3377
Ø7ØØ – Ø7FF	PAGE 7 (RAM) REGISTER STORAGE & STACK	ØØ34ØØ – ØØ3777
Ø8ØØ – FFFF	NOT USED	ØØ4ØØØ – 177777

Page 7 locations used by monitor program

OCTAL/HEX LOCATION	CONTENTS	
267/B7	KYTEM (current value	
271/B9	LVALU of display)	
272/BA	HVALU	
273/BB	PCL	PCSTO
274/BC	PCH	PCSTO
275/BD	PSW	PSWST
276/BE	A REG	PSWST
277/BF	C REG	BSTOR
3ØØ/CØ	B REG	BSTOR
3Ø1/C1	E REG	DSTOR
3Ø2/C2	D REG	DSTOR
3Ø3/C3	L REG	HSTOR
3Ø4/C4	H REG	HSTOR
3Ø5/C5	OFLAG	

36Ø/FØ STACK PTR

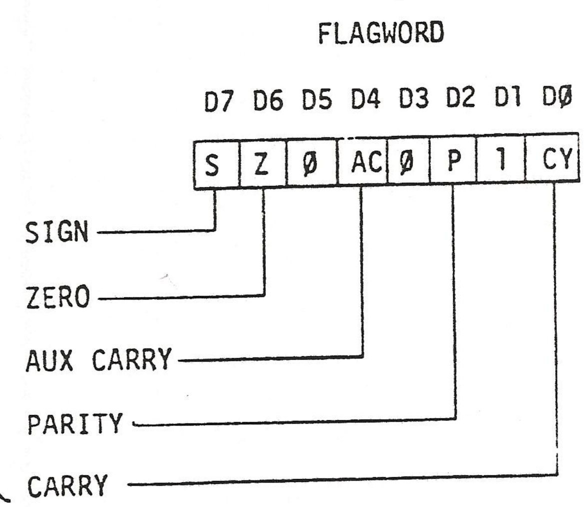


Figure 3. Memory map for MST-80B Microcomputer trainer.

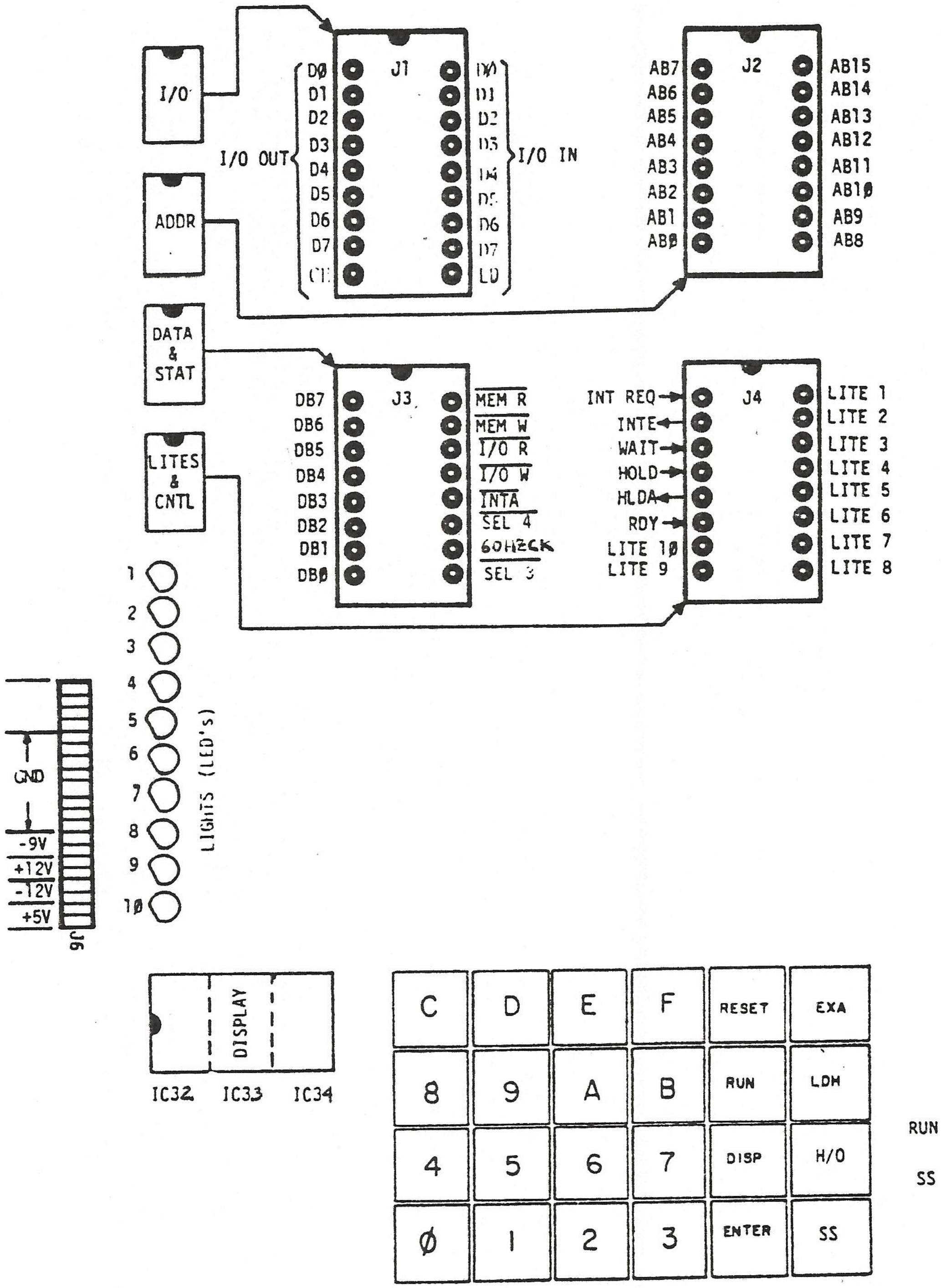


Figure 4 Panel connectors used to interface MST-80B microcomputer trainer.

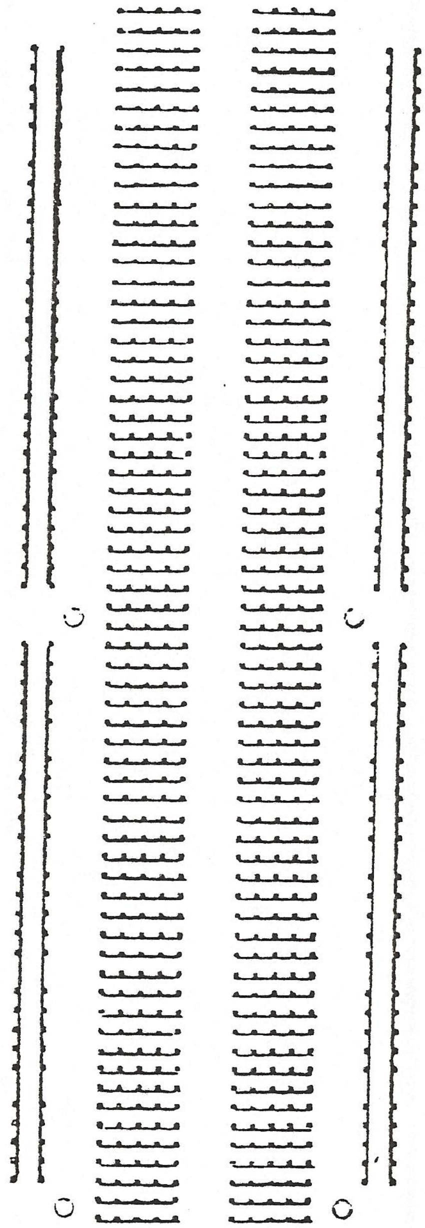


Figure 5 Breadboard Hidden Connections

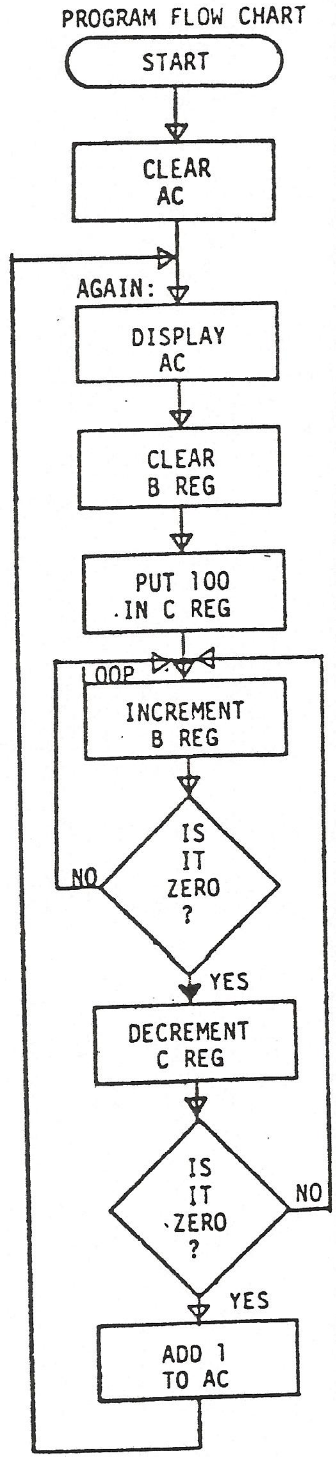


Figure 6 Flow chart for BREAK POINT example program for MST-80B.

APPENDIX

TABLE 1   8080 INSTRUCTION SET

Mnemonic	Description	Code	#	~	C	AC	Z	S	P
MOVE, LOAD, STORE									
MOV rd,rs	Move reg.s to reg.d	2ds	1	5	-	-	-	-	-
MOV M,rs	Move reg.s to (M)	26s	1	7	-	-	-	-	-
MOV rd,M	Move (M) to reg.d	2d6	1	7	-	-	-	-	-
MVI r,data	Move immed. to reg.	0d6	2	7	-	-	-	-	-
MVI M,data	Move immed. to (M)	36	2	10	-	-	-	-	-
LXI B,data	Load immed. reg. B-C	C1	3	10	-	-	-	-	-
LXI D,data	Load immed. reg. D-E	11	3	10	-	-	-	-	-
LXI H,data	Load immed. reg. H-L	21	3	10	-	-	-	-	-
STAX B	Store A at (B-C)	02	1	7	-	-	-	-	-
STAX D	Store A at (D-E)	12	1	7	-	-	-	-	-
LDAX B	Load A at (B-C)	0A	1	7	-	-	-	-	-
LDAX D	Load A at (D-E)	1A	1	7	-	-	-	-	-
STA addr	Store A direct	32	3	13	-	-	-	-	-
LDA addr	Load A direct	3A	3	13	-	-	-	-	-
SHLD addr	Store H-L direct	22	3	16	-	-	-	-	-
LHLD addr	Load H-L direct	2A	3	16	-	-	-	-	-
XCHG	Exchange D-E and H-L	EB	1	4	-	-	-	-	-
STACK OPS									
PUSH B	Push B-C	C5	1	11	-	-	-	-	-
PUSH D	Push D-E	D5	1	11	-	-	-	-	-
PUSH H	Push H-L	E5	1	11	-	-	-	-	-
PUSH PSW	Push A and Flags	F5	1	11	-	-	-	-	-
POP B	Pop B-C	C1	1	10	-	-	-	-	-
POP D	Pop D-E	D1	1	10	-	-	-	-	-
POP H	Pop H-L	E1	1	10	-	-	-	-	-
POP PSW	Pop A and Flags	F1	1	10	-	-	-	-	-
XTHL	Exchange stack & H-L	E3	1	18	-	-	-	-	-
SPHL	Load SP with H-L	F9	1	5	-	-	-	-	-
LXI SP,data	Load immed. SP	31	3	10	-	-	-	-	-
INX SP	Increment SP	33	1	5	-	-	-	-	-
DCX SP	Decrement SP	3B	1	5	-	-	-	-	-
JUMPS									
JMP addr	Jump unconditionally	C3	3	10	-	-	-	-	-
JC addr	Jump on carry	DA	3	10	-	-	-	-	-
JNC addr	Jump on no carry	D2	3	10	-	-	-	-	-
JZ addr	Jump on zero	CA	3	10	-	-	-	-	-
JNZ addr	Jump on no zero	C2	3	10	-	-	-	-	-
JP addr	Jump on positive	F2	3	10	-	-	-	-	-
JM addr	Jump on minus	FA	3	10	-	-	-	-	-
JPE addr	Jump on parity even	EA	3	10	-	-	-	-	-
JPO addr	Jump on parity odd	E2	3	10	-	-	-	-	-
PCHL	Load PC with H-L	E9	1	5	-	-	-	-	-
RST n	Restart @ loc. n*2**3	3n7	1	11	-	-	-	-	-
CALLS									
CALL addr	Call unconditionally	CD	3	17	-	-	-	-	-
CC addr	Call on carry	DC	3	17	-	-	-	-	-
CNC addr	Call on no carry	D4	3	17	-	-	-	-	-
CZ addr	Call on zero	CC	3	17	-	-	-	-	-
CNZ addr	Call on no zero	C4	3	17	-	-	-	-	-
CP addr	Call on positive	F4	3	17	-	-	-	-	-
CM addr	Call on minus	FC	3	17	-	-	-	-	-
CPE addr	Call on parity even	EC	3	17	-	-	-	-	-
CPO addr	Call on parity odd	F4	3	17	-	-	-	-	-

Mnemonic	Description	Code	#	~	C	AC	Z	S	P
RETURNS									
RET	Return unconditionally	C9	1	10	-	-	-	-	-
RC	Return on carry	D8	1	11	-	-	-	-	-
RNC	Return on no carry	D0	1	11	-	-	-	-	-
RZ	Return on zero	C8	1	11	-	-	-	-	-
RNZ	Return on no zero	C0	1	11	-	-	-	-	-
RP	Return on positive	F0	1	11	-	-	-	-	-
RM	Return on minus	F8	1	11	-	-	-	-	-
RPE	Return on parity even	EC	1	11	-	-	-	-	-
RPO	Return on parity odd	E4	1	11	-	-	-	-	-
ADDS									
ADD r	Add r to A	20s	1	4	x	x	x	x	x
ADC r	Add r to A +CY	21s	1	4	x	x	x	x	x
ADD M	Add (M) to A	86	1	7	x	x	x	x	x
ADC M	Add (M) to A +CY	8E	1	7	x	x	x	x	x
ADI data	Add immed. to A	C6	2	7	x	x	x	x	x
ACI data	Add immed. to A +CY	CE	2	7	x	x	x	x	x
DAD B	Add B-C to H-L	09	1	10	x	-	-	-	-
DAD D	Add D-E to H-L	19	1	10	x	-	-	-	-
DAD H	Add H-L to H-L	29	1	10	x	-	-	-	-
DAD SP	Add SP to H-L	39	1	10	x	-	-	-	-
SUBTRACTS									
SUB r	Subtract r from A	22s	1	4	x	x	x	x	x
SBB r	Subtract r from A -CY	23s	1	4	x	x	x	x	x
SUB M	Sub. (M) from A	96	1	7	x	x	x	x	x
SBB M	Sub. (M) from A -CY	9E	1	7	x	x	x	x	x
SUI data	Sub. immed. from A	D6	2	7	x	x	x	x	x
SBI data	Sub. immed. from A -CY	DE	2	7	x	x	x	x	x
CMP r	Compare: A-r	27s	1	4	x	x	x	x	x
CMP M	Compare: A-(M)	BE	1	7	x	x	x	x	x
CPI data	Compare: A-data	FE	2	7	x	x	x	x	x
LOGICALS									
ANA r	AND r with A	24s	1	4	x	x	x	x	x
XRA r	XOR r with A	25s	1	4	x	x	x	x	x
ORA r	OR r with A	26s	1	4	x	x	x	x	x
ANA M	AND (M) with A	A6	1	7	x	x	x	x	x
XRA M	XOR (M) with A	AE	1	7	x	x	x	x	x
ORA M	OR (M) with A	B6	1	7	x	x	x	x	x
ANI data	AND immed. with A	E6	2	7	x	x	x	x	x
XRI data	XOR immed. with A	EE	2	7	x	x	x	x	x
ORI data	OR immed.with A	F6	2	7	x	x	x	x	x
INCREMENTS AND DECREMENTS									
INR r	Increment r	0d4	1	5	-	x	x	x	x
DCR r	Decrement r	0d5	1	5	-	x	x	x	x
INR M	Increment (M)	34	1	10	-	x	x	x	x
DCR M	Decrement (M)	35	1	10	-	x	x	x	x
INX B	Increment B-C	03	1	5	-	-	-	-	-
INX D	Increment D-E	13	1	5	-	-	-	-	-
INX H	Increment H-L	23	1	5	-	-	-	-	-
DCX B	Decrement B-C	0B	1	5	-	-	-	-	-
DCX D	Decrement D-E	1B	1	5	-	-	-	-	-
DCX H	Decrement H-L	2B	1	5	-	-	-	-	-

Mnemonic	Description	Code	#	~	C	AC	Z	S	P
ROTATES									
RLC	Rotate A left	07	1	4	x	-	-	-	-
RRC	Rotate A right	0F	1	4	x	-	-	-	-
RAL	Rotate A & carry left	27	1	4	x	-	-	-	-
RAR	Rotate A & carry right	1F	1	4	x	-	-	-	-
SPECIALS									
CMA	Complement A	2F	1	4	-	-	-	-	-
STC	Set carry	37	1	4	1	-	-	-	-
CMC	Complement carry	3F	1	4	x	-	-	-	-
DAA	Decimal adjust A	27	1	4	x	x	x	x	x
INPUT/OUTPUT									
IN n	Input port n to A	DB	2	10	-	-	-	-	-
OUT n	Output A to port n	D3	2	10	-	-	-	-	-
CONTROL									
EI	Enable interrupt	FB	1	4	-	-	-	-	-
DI	Disable interrupt	F3	1	4	-	-	-	-	-
NOP	No operation	00	1	4	-	-	-	-	-
HLT	Halt execution	76	1	7	-	-	-	-	-

\# - number of bytes/instruction
~ - number of clock cycles/instruction
C - carry flag
AC- auxiliary or half carry flag
Z - zero flag
S - sign flag (set if minus)
P - parity flag (set if even)

NOTES: 1. Register codes for r,rs,rd are
B-1
C-2
D-3
E-4
H-5
L-6
A-7

2. Execution times for conditions not met are 11 usec for CALLS and 5 usec for RETURNS. Conditional jumps all execute in same time, condition true or false.

TABLE 2

8080 ASSEMBLY LANGUAGE REFERENCE CARD

ALPHABETICAL LISTING

OCT	HEX	MNEMONIC	OCT	HEX	MNEMONIC	OCT	HEX	MNEMONIC	OCT	HEX	MNEMONIC	OCT	HEX	MNEMONIC
316	CE	ACI D8	71	39	DAD SP	174	7C	MOV A,H	167	77	MOV M,A	347	E7	RST 4
217	8F	ADC A	75	3D	DCR A	175	7D	MOV A,L	160	70	MOV M,B	357	EF	RST 5
210	88	ADC B	05	05	DCR B	176	7E	MOV A,M	161	71	MOV M,C	367	F7	RST 6
211	89	ADC C	15	0D	DCR C	107	47	MOV B,A	162	72	MOV M,D	377	FF	RST 7
212	8A	ADC D	25	15	DCR D	100	40	MOV B,B	163	73	MOV M,E	310	C8	RZ
213	8B	ADC E	35	1D	DCR E	101	41	MOV B,C	164	74	MOV M,H	327	9F	SBB A
214	8C	ADC H	45	25	DCR H	102	42	MOV B,D	165	75	MOV M,L	230	98	SBB B
215	8D	ADC L	55	2D	DCR L	103	43	MOV B,E	76	3E	MVI A,D8	231	99	SBB C
216	8E	ADC M	65	35	DCR M	104	44	MOV B,H	06	06	MVI B,D8	232	9A	SBB D
207	87	ADD A	13	0B	DCX B	105	45	MOV B,L	16	0E	MVI C,D8	233	9B	SBB E
200	80	ADD B	33	1B	DCX D	106	46	MOV B,M	26	16	MVI D,D8	234	9C	SBB H
201	81	ADD C	53	2B	DCX H	117	4F	MOV C,A	36	1E	MVI E,D8	235	9D	SBB L
202	82	ADD D	73	3B	DCX SP	110	48	MOV C,B	46	26	MVI H,D8	236	9E	SBB M
203	83	ADD E	363	F3	DI	111	49	MOV C,C	56	2E	MVI L,D8	336	DE	SBI D8
204	84	ADD H	373	FB	EI	112	4A	MOV C,D	66	36	MVI M,D8	42	22	SHLD Adr
205	85	ADD L	166	76	HLT	113	4B	MOV C,E	00	00	NOP	371	F9	SPHL
206	86	ADD M	333	DB	IN D8	114	4C	MOV C,H	267	B7	ORA A	62	32	STA Adr
306	C6	ADI D8	74	3C	INR A	115	4D	MOV C,L	260	B0	ORA B	02	02	STAX B
247	A7	ANA A	04	04	INR B	116	4E	MOV C,M	261	B1	ORA C	22	12	STAX D
240	A0	ANA B	14	0C	INR C	127	57	MOV D,A	262	B2	ORA D	67	37	STC
241	A1	ANA C	24	14	INR D	120	50	MOV D,B	263	B3	ORA E	227	97	SUB A
242	A2	ANA D	34	1C	INR E	121	51	MOV D,C	264	B4	ORA H	220	90	SUB B
243	A3	ANA E	44	24	INR H	122	52	MOV D,D	265	B5	ORA L	221	91	SUB C
244	A4	ANA H	54	2C	INR L	123	53	MOV D,E	266	B6	ORA M	222	92	SUB D
245	A5	ANA L	64	34	INR M	124	54	MOV D,H	366	F6	ORI D8	223	93	SUB E
246	A6	ANA M	03	03	INX B	125	55	MOV D,L	323	D3	OUT D8	224	94	SUB H
346	E6	ANI D8	23	13	INX D	126	56	MOV D,M	351	E9	PCHL	225	95	SUB L
315	CD	CALL Adr	43	23	INX H	137	5F	MOV E,A	301	C1	POP B	226	96	SUB M
334	DC	CC Adr	63	33	INX SP	130	58	MOV E,B	321	D1	POP D	326	D6	SUI D8
374	FC	CM Adr	332	DA	JC Adr	131	59	MOV E,C	341	E1	POP H	353	EB	XCHG
57	2F	CMA	372	FA	JM Adr	132	5A	MOV E,D	361	F1	POP PSW	257	AF	XRA A
77	3F	CMC	303	C3	JMP Adr	133	5B	MOV E,E	305	C5	PUSH B	250	A8	XRA B
277	BF	CMP A	322	D2	JNC Adr	134	5C	MOV E,H	325	D5	PUSH D	251	A9	XRA C
270	B8	CMP B	302	C2	JNZ Adr	135	5D	MOV E,L	345	E5	PUSH H	252	AA	XRA D
271	B9	CMP C	362	F2	JP Adr	136	5E	MOV E,M	365	F5	PUSH PSW	253	AB	XRA E
272	BA	CMP D	352	EA	JPE Adr	147	67	MOV H,A	27	17	RAL	254	AC	XRA H
273	BB	CMP E	342	E2	JPO Adr	140	60	MOV H,B	37	1F	RAR	255	AD	XRA L
274	BC	CMP H	312	CA	JZ	141	61	MOV H,C	330	D8	RC	256	AE	XRA M
275	BD	CMP L	72	3A	LDA Adr	142	62	MOV H,D	311	C9	RET	356	EE	XRI D8
276	BE	CMP M	12	0A	LDAX B	143	63	MOV H,E	07	07	RLC	343	E3	XTHL
324	D4	CNC Adr	32	1A	LDAX D	144	64	MOV H,H	370	F8	RM	10	08	---
304	C4	CNZ Adr	52	2A	LHLD Adr	145	65	MOV H,L	320	D0	RNC	20	10	---
364	F4	CP Adr	01	01	LXI B,D16	146	66	MOV H,M	300	C0	RNZ	30	18	---
354	EC	CPE Adr	21	11	LXI D,D16	157	6F	MOV L,A	360	F0	RP	40	20	---
376	FE	CPI D8	41	21	LXI H,D16	150	68	MOV L,B	350	E8	RPE	50	28	---
344	E4	CPO Adr	61	31	LXI SP,D16	151	69	MOV L,C	340	E0	RPO	60	30	---
314	CC	CZ Adr	177	7F	MOV A,A	152	6A	MOV L,D	17	0F	RRC	70	38	---
47	27	DAA	170	78	MOV A,B	153	6B	MOV L,E	307	C7	RST 0	313	CB	---
11	09	DAD B	171	79	MOV A,C	154	6C	MOV L,H	317	CF	RST 1	331	D9	---
31	19	DAD D	172	7A	MOV A,D	155	6D	MOV L,L	327	D7	RST 2	335	DD	---
51	29	DAD H	173	7B	MOV A,E	156	6E	MOV L,M	337	DF	RST 3	355	ED	---
												375	FD	---

D8 = constant, or expression that evaluates to an 8 bit data quantity.

D16 = constant, or expression that evaluates to a 16 bit data quantity.

Adr = 16 bit address.

TABLE 3

8080 ASSEMBLY LANGUAGE REFERENCE CARD

NUMERICAL LISTING

OCT	HEX	MNEMONIC
00	00	NOP
01	01	LXI B,D16
02	02	STAX B
03	03	INX B
04	04	INR B
05	05	DCR B
06	06	MVI B,D8
07	07	RLC
10	08	---
11	09	DAD B
12	0A	LDAX B
13	0B	DCX B
14	0C	INR C
15	0D	DCR C
16	0E	MVI C,D8
17	0F	RRC
20	10	---
21	11	LXI D,D16
22	12	STAX D
23	13	INX D
24	14	INR D
25	15	DCR D
26	16	MVI D,D8
27	17	RAL
30	18	---
31	19	DAD D
32	1A	LDAX D
33	1B	DCX D
34	1C	INR E
35	1D	DCR E
36	1E	MVI E,D8
37	1F	RAR
40	20	---
41	21	LXI H,D16
42	22	SHLD Adr
43	23	INX H
44	24	INR H
45	25	DCR H
46	26	MVI H,D8
47	27	DAA
50	28	---
51	29	DAD H
52	2A	LHLD Adr
53	2B	DCX H
54	2C	INR L
55	2D	DCR L
56	2E	MVI L,D8
57	2F	CMA
60	30	---
61	31	LXI SP,D16
62	32	STA Adr
63	33	INX SP
64	34	INR M
65	35	DCR M
66	36	MVI M,D8
67	37	STC
70	38	---
71	39	DAD SP
72	3A	LDA Adr
73	3B	DCX SP
74	3C	INR A
75	3D	DCR A
76	3E	MVI A,D8
77	3F	CMC
100	40	MOV B,B
101	41	MOV B,C
102	42	MOV B,D
103	43	MOV B,E
104	44	MOV B,H
105	45	MOV B,L
106	46	MOV B,M
107	47	MOV B,A
110	48	MOV C,B
111	49	MOV C,C
112	4A	MOV C,D
113	4B	MOV C,E
114	4C	MOV C,H
115	4D	MOV C,L
116	4E	MOV C,M
117	4F	MOV C,A
120	50	MOV D,B
121	51	MOV D,C
122	52	MOV D,D
123	53	MOV D,E
124	54	MOV D,H
125	55	MOV D,L
126	56	MOV D,M
127	57	MOV D,A
130	58	MOV E,B
131	59	MOV E,C
132	5A	MOV E,D
133	5B	MOV E,E
134	5C	MOV E,H
135	5D	MOV E,L
136	5E	MOV E,M
137	5F	MOV E,A
140	60	MOV H,B
141	61	MOV H,C
142	62	MOV H,D
143	63	MOV H,E
144	64	MOV H,H
145	65	MOV H,L
146	66	MOV H,M
147	67	MOV H,A
150	68	MOV L,B
151	69	MOV L,C
152	6A	MOV L,D
153	6B	MOV L,E
154	6C	MOV L,H
155	6D	MOV L,L
156	6E	MOV L,M
157	6F	MOV L,A
160	70	MOV M,B
161	71	MOV M,C
162	72	MOV M,D
163	73	MOV M,E
164	74	MOV M,H
165	75	MOV M,L
166	76	HLT
167	77	MOV M,A
170	78	MOV A,B
171	79	MOV A,C
172	7A	MOV A,D
173	7B	MOV A,E
174	7C	MOV A,H
175	7D	MOV A,L
176	7E	MOV A,M
177	7F	MOV A,A
200	80	ADD B
201	81	ADD C
202	82	ADD D
203	83	ADD E
204	84	ADD H
205	85	ADD L
206	86	ADD M
207	87	ADD A
210	88	ADC B
211	89	ADC C
212	8A	ADC D
213	8B	ADC E
214	8C	ADC H
215	8D	ADC L
216	8E	ADC M
217	8F	ADC A
220	90	SUB B
221	91	SUB C
222	92	SUB D
223	93	SUB E
224	94	SUB H
225	95	SUB L
226	96	SUB M
227	97	SUB A
230	98	SBB B
231	99	SBB C
232	9A	SBB D
233	9B	SBB E
234	9C	SBB H
235	9D	SBB L
236	9E	SBB M
237	9F	SBB A
240	A0	ANA B
241	A1	ANA C
242	A2	ANA D
243	A3	ANA E
244	A4	ANA H
245	A5	ANA L
246	A6	ANA M
247	A7	ANA A
250	A8	XRA B
251	A9	XRA C
252	AA	XRA D
253	AB	XRA E
254	AC	XRA H
255	AD	XRA L
256	AE	XRA M
257	AF	XRA A
260	B0	ORA B
261	B1	ORA C
262	B2	ORA D
263	B3	ORA E
264	B4	ORA H
265	B5	ORA L
266	B6	ORA M
267	B7	ORA A
270	B8	CMP B
271	B9	CMP C
272	BA	CMP D
273	BB	CMP E
274	BC	CMP H
275	BD	CMP L
276	BE	CMP M
277	BF	CMP A
300	C0	RNZ
301	C1	POP B
302	C2	JNZ Adr
303	C3	JMP Adr
304	C4	CNZ Adr
305	C5	PUSH B
306	C6	ADI D8
307	C7	RST 0
310	C8	RZ
311	C9	RET
312	CA	JZ
313	CB	---
314	CC	CZ Adr
315	CD	CALL Adr
316	CE	ACI D8
317	CF	RST 1
320	D0	RNC
321	D1	POP D
322	D2	JNC Adr
323	D3	OUT D8
324	D4	CNC Adr
325	D5	PUSH D
326	D6	SUI D8
327	D7	RST 2
330	D8	RC
331	D9	---
332	DA	JC Adr
333	DB	IN D8
334	DC	CC Adr
335	DD	---
336	DE	SBI D8
337	DF	RST 3
340	E0	RPO
341	E1	POP H
342	E2	JPO Adr
343	E3	XTHL
344	E4	CPO Adr
345	E5	PUSH H
346	E6	ANI D8
347	E7	RST 4
350	E8	RPE
351	E9	PCHL
352	EA	JPE Adr
353	EB	XCHG
354	EC	CPE Adr
355	ED	---
356	EE	XRI D8
357	EF	RST 5
360	F0	RP
361	F1	POP PSW
362	F2	JP Adr
363	F3	DI
364	F4	CP Adr
365	F5	PUSH PSW
366	F6	ORI D8
367	F7	RST 6
370	F8	RM
371	F9	SPHL
372	FA	JM Adr
373	FB	EI
374	FC	CM Adr
375	FD	---
376	FE	CPI D8
377	FF	RST 7

D8 = constant, or expression that evaluates to an 8 bit data quantity.

D16 = constant, or expression that evaluates to a 16 bit data quantity.

Adr - 16 bit address.

Program Listing, MST-80B Microcomputer Monitor Program

8080 MACRO ASSEMBLER, VER 2.2 ERRORS = 0 PAGE 1

```
                                                ;+++++++++HEX/OCT MONITOR+++++++++
                                           ;++++++FOR MST-80 MICROPROCESSOR TRAINER++++++

                                                ;WRITTEN BY GORDON JONES
                                                ;DATE: 8-23-76
                                 ;
                                 ;
                                 ;
003667                           KYTEM    EQU    07B7H
003671                           LVALU    EQU    07B9H
003672                           HVALU    EQU    07BAH
003673                           PCSTO    EQU    07BBH
003675                           PSWST    EQU    07BDH
003677                           BSTOR    EQU    07BFH
003701                           DSTOR    EQU    07C1H
003703                           HSTOR    EQU    07C3H
003705                           OFLAG    EQU    HSTOR+2
                                 ;
                                 ;
                                 ;
002407                           KEYBD    EQU    0507H
002401                           KYBD1    EQU    0501H
                                 ;
000360                           TOP      EQU    0F0H
000017                           BOT      EQU    0FH
000002                           RREAD    EQU    2H
003673                           BKSTO    EQU    07BBH
000006                           DISO     EQU    6
000007                           DISH     EQU    7

                                          ;+++++++INTIALIZE ROUTINE+++++++
                                          ORG    0       Fφ
00000   061 360 007     INIT:     LXI    SP,070FH       ;INIT STACK POINTER
00003   257                       XRA    A              ;CLEAR ACCUMULATOR
00004   062 267 007               STA    KEYTEM         ;INIT DISPLAY SAVE
00007   062 305 007               STA    OFLAG          ;SET HEX DISP. MODE
00012   323 007                   OUT    DISH           ;SET TO DISP. IN HEX
00014   315 117 001               CALL   DIS            ;CLEAR DISPLAY
00017   315 131 000     ST:       CALL   KEY            ;GO TO KEY ROUTINE
00022   303 017 000               JMP    ST             ;AWAIT A COMMAND
00025   316             TABLC:    DB     ENTER          ;CONTROL ROUTINE ADDRS.
00026   270                       DB     DISP
00027   234                       DB     RUN
00030   131                       DB     KEY
00031   131                       DB     KEY
00032   331                       DB     HO
00033   223                       DB     LDH
00034   215                       DB     EXA

00040   303 060 007               JMP    0730H
00050   303 120 007               JMP    0750H
00060   042 303 007               SHLD   07C3H          ;BREAK POINT ENTRY
00063   303 073 000               JMP    BREAK
00070   303 000 007               JMP    0700H          ;RST7 INTERRUPT ENTRY
```

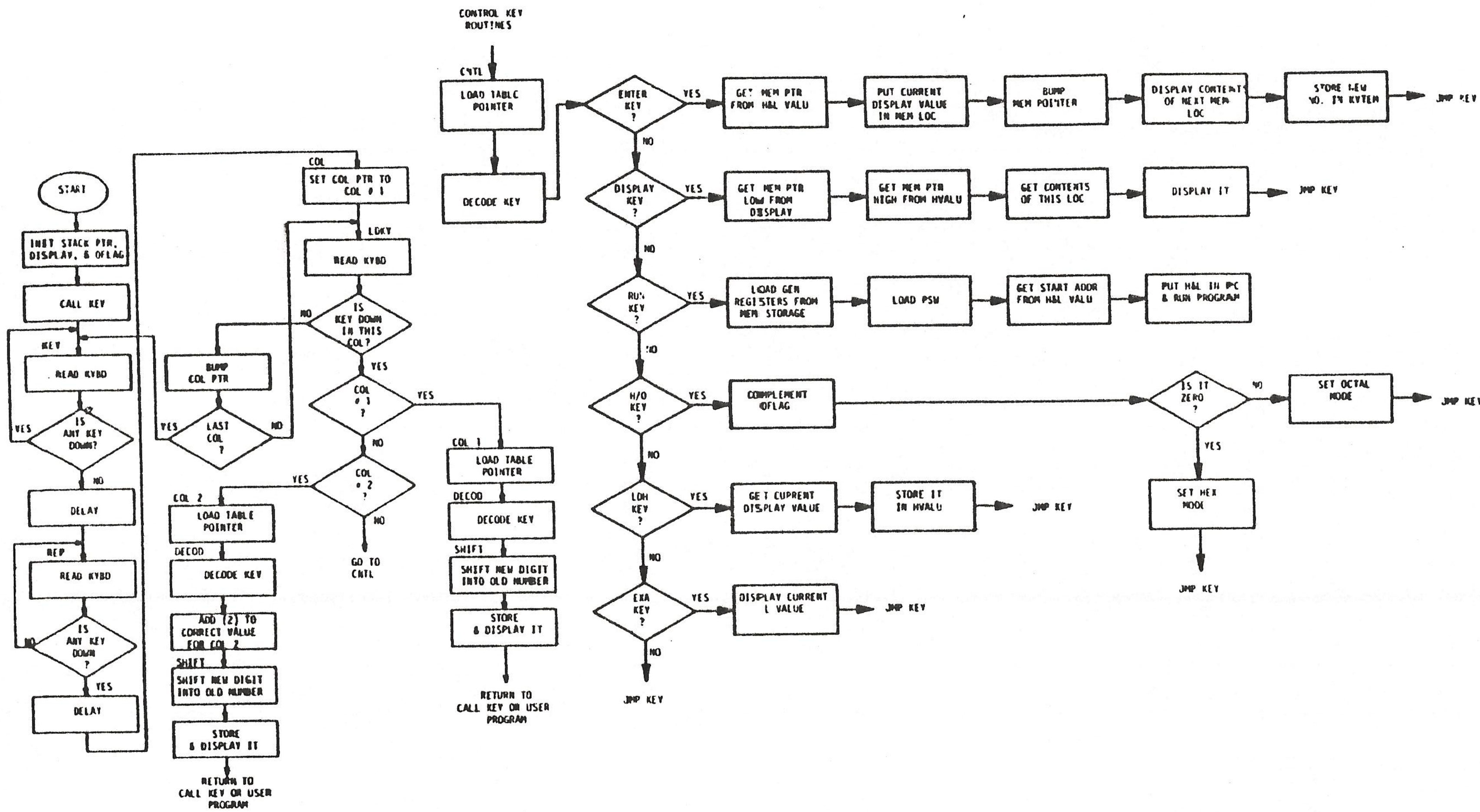
```
                              ;*******THIS IS THE BREAK ROUTINE*****

                        BRK:
000073   341                         POP    H              ;PUT BREAK ADDRESS IN H&L REG
000074   053                         DCX    H              ;CORRECT BRK ADDR
000075   042 273 007                 SHLD   PCSTOR         ;STORE BREAK ADDR IN MEMORY
000100   365                         PUSH   PSW            ;GET AC AND PSW IN STACK
000101   341                         POP    H              ;PUT AC &PSW IN H&L
000102   042 275 007                 SHLD   PSWST          ;PUT AC &PSW IN MEMORY
000105   305                         PUSH   B              ;GET B&C
000106   341                         POP    H              ;PUT B&C IN MEMORY
000107   042 277 007                 SHLD   BSTOR          ;PUT B&C IN MEMORY
000112   353                         XCHG                  ;PUT D&E IN H&L
000113   042 301 007                 SHLD   DSTOR          ;PUT D&E IN MEMORY
000116   041 273 007                 LXI    H,BKSTO        ;LOAD BREAK MEMORY LOCATION
000121   042 271 007                 SHLD   LVALU          ;PUT IT IN PROPER LOCATION
000124   076 273                     MVI    A,0BBH         ;PUT BB IN AC
000126   303 305 000                 JMP    BACK           ;DISPLAY BB AND RETURN TO KEY

                                   ;*****KEYBOARD READ ROUTINE*****

000131   315 111 001    KEY:         CALL   READ           ;GO READ KEYBOARD
000134   302 131 000                 JNZ    KEY            ;LOOP IF KEY DOWN
000137   315 161 001                 CALL   DELAY          ;DEBOUNCE

000142   315 117 001    REP:         CALL   DIS            ;CHECK FOR CHANGE IN DISP MODE
000145   315 111 001                 CALL   READ           ;GO READ KEYBOARD
000150   312 142 000                 JZ     REP            ;LOOP IF NO KEY DOWN
000153   315 161 001                 CALL   DELAY          ;DEBOUNCE
000156   041 001 005    COL:         LXI    H,KYBD1        ;SET UP COLUMN POINTER
000161   176            LDKY:        MOV    A,M            ;READ KEYBOARD COLUMN
000162   057                         CMA                   ;COMPLEMENT
000163   267                         ORA    A              ;SET FLAGS
000164   302 356 000                 JNZ    LUT            ;GOTO LOOK UP TABLE IF KEY FOUND
000167   175                         MOV    A,L            ;NO KEY FOUND - BUMP COLUMN POINTER
000170   027                         RAL                   ;ROTATE TO NEXT COLUMN
000171   157                         MOV    L,A            ;PUT BACK
000172   346 010                     ANI    08H            ;CHECK FOR LAST COLUMN
000174   312 161 000                 JZ     LDKY           ;NOT LAST COLUMN - GO READ A KEY
000177   303 131 000                 JMP    KEY            ;NO KEY DOWN GO BACK

                               ;********THESE ARE THE CONTROL KEY ROUTINES

000202   041 022 000    CNTL:        LXI    H,TABLE-1      ;GET TABLE POINTER
```

```
000205  170                           MOV   A,B       ;GET KEY VALUE
000206  027             LP1:          RAL             ;ROTATE INTO CARRY
000207  043                           INX   H         ;BUMP TABLE POINTER
000210  322 206 000                   JNC   LP1       ;
000213  156                           MOV   L,M       ;MOVE ADDRESS INTO L REG
000214  351                           PCHL            ;JUMP TO PROPER CONTROL ROUTINE
000215  072 271 007     EXA:          LDA   LVALU     ;GET L REGISTER VALUE
000220  303 305 000                   JMP   BACK      ;DISPLAY IT & JUMP TO KEY
000223  072 267 007     LDH:          LDA   KYTEM     ;GET KEY VALUE FROM TEMP
000226  062 272 007                   STA   HVALU     ;PUT IN H REGISTER STORAGE
000231  303 131 000                   JMP   KEY       ;DONE- GO TO START


000234  072 267 007     RUN:          LDA   KYTEM     ;GET CURRENT DISPLAY VALUE
000237  062 271 007                   STA   LVALU     ;STORE IN L REG LOCATION
000242  052 277 007                   LHLD  BSTOR     ;GET CONTENTS OF B&C REGS
000245  345                           PUSH  H         ;PUT ON STACK
000246  301                           POP   B         ;PUT IN B&C REGS
000247  052 301 007                   LHLD  DSTOR     ;GET CONTENTS OF D&E REGS
000252  353                           XCHG            ;EXCHANGE H&L WITH D&E
000253  052 275 007                   LHLD  PSWST     ;GET OLD AC AND PSW
000256  345                           PUSH  H         ;PUT AC & PSW ON STACK
000257  361                           POP   PSW       ;RESTORE AC & STATUS
000260  052 271 007                   LHLD  LVALU     ;GET STARTING ADDRESS
000263  345                           PUSH  H         ;PUT STARTING ADDR ON STACK
000264  052 303 007                   LHLD  HSTOR     ;RESTORE H&L
000267  311                           RET             ;GET STARTING ADDR FROM STACK AND RUN

000270  072 267 007     DISP:         LDA   KYTEM     ;GET CURRENT DISPLAY VALUE
000273  062 271 007                   STA   LVALU     ;STORE IN LREG STORAGE
000276  052 271 007                   LHLD  LVALU     ;GET VALUE JUST KEYED IN
000301  042 271 007     NEXT:         SHLD  LVALU     ;STORE IN MEMORY POINTER
000304  176                           MOV   A,M       ;GET VALUE POINTED TO BY MEM POINTER
000305  062 267 007     BACK:         STA   KYTEM     ;PUT THIS VALUE IN KEY STORAGE
000310  315 117 001                   CALL  DIS       ;DISPLAY IT
000313  303 131 000                   JMP   KEY       ;GO BACK AND START OVER
000316  052 271 007     ENTER:        LHLD  LVALU     ;GET MEMORY POINTER
000321  072 267 007                   LDA   KYTEM     ;GET DISPLAY VALUE
000324  167                           MOV   M,A       ;PUT VALUE IN LOC POINTED TO BY H&L
000325  043                           INX   H         ;BUMP TO NEXT LOCATION
000326  303 301 000                   JMP   NEXT      ;PUT INC PTR AWAY AND DISPLAY NEXT LOC

000331  072 305 007     HO:           LDA   OFLAG     ;FETCH HEX/OCTAL FLAG
000334  057                           CMA             ;CHANGE TO OTHER BASE
000335  062 305 007                   STA   OFLAG     ;PUT IT BACK
000340  267                           ORA   A         ;SET-UP FOR TESTING IT
000341  312 351 000                   JZ    HOHO      ;JMP IF 0 FOR HEX
000344  323 006                       OUT   DISO      ;MUST BE 1'S FOR OCTAL - SET DISPLAY
000346  303 131 000                   JMP   KEY
```

```
000351  323 007        HOHO:  OUT   DISH          ;SET DISPLAY FOR HEX 3 DIGITS
000353  303 131 000           JMP   KEY


                       ;********THIS ROUTINE DETERMINES THE COLUMN
                       ;*********THE KEY WAS FOUND IN AND LOOKS UP
                       ;*********VALUE IN THE APPROPRIATE TABLE.

000356  107            LUI:   MOV   B,A           ;SAVE AC
000357  175                   MOV   A,L           ;GET COLUMN POINTER
000360  017                   RRC                 ;ROTATE COL POINTER RIGHT
000361  332 373 000           JC    COL1          ;IS IT COL1?
000364  017                   RRC                 ;ROTATE AGAIN
000365  332 004 001           JC    COL2          ;IS IT COL2?
000370  303 202 000           JMP   CNTL          ;MUST BE CONTROL COLUMN

000373  041 171 001    COL1:  LXI   H,TABLE-1     ;GET TABLE POINTER
000376  315 063 001           CALL  DECOD         ;GO GET VALUE FROM TABLE
000401  303 016 001           JMP   SHIFT         ;STORE AND SEND TO DISPLAY
000404  041 171 001    COL2:  LXI   H,TABLE-1     ;GET TABLE POINTER
000407  315 063 001           CALL  DECOD         ;GET VALUE FROM TABLE
000412  171                   MOV   A,C           ;PUT TABLE VALUE IN AC
000413  306 002               ADI   2H            ;CORRECT VALUE FOR COLUMN 2
000415  117                   MOV   C,A
000416  041 267 007    SHIFT: LXI   H,KYTEM       ;GET OLD DISPLAY VALUE
000421  000                   NOP
000422  000                   NOP
000423  072 305 007           LDA   OFLAG         ;CHECK HEX/OCT FLAG
000426  267                   ORA   A             ;SET FLAGS
000427  302 047 001           JNZ   OCT1          ;GOTO OCTAL IF FLAG IS A 1

000432  176                   MOV   A,M           ;GET KEY CODE
000433  007            HEX1:  RLC                 ;ROTATE ONE HEX DIGIT LEFT
000434  007                   RLC
000435  007                   RLC
000436  007                   RLC
000437  346 360               ANI   0F0H          ;MASK OFF BOTTOM DIGIT
000441  261                   ORA   C             ;OR NEW DIGIT TO OLD NUMBER
000442  167                   MOV   M,A           ;PUT BACK IN DISPLAY STORAGE
000443  315 117 001           CALL  DIS           ;SEND TO DISPLAY
000446  311                   RET                 ;END OF NUMBER KEY ROUTINE

000447  176            OCT1:  MOV   A,M           ;GET KEY CODE
000450  007                   RLC                 ;ROTATE ONE OCTAL DIGIT LEFT
000451  007                   RLC
000452  007                   RLC
000453  346 370               ANI   370Q          ;MASK OFF BOTTOM DIGIT
000455  261                   ORA   C             ;OR NEW DIGIT TO OLD NUMBER
000456  167                   MOV   M,A           ;PUT BACK IN DISPLAY STORAGE
```

```
000457  315 117 001            CALL   DIS        ;SEND TO DISPLAY
000462  311                    RET

000463  170          DECOD:    MOV    A,B        ;GET KEY VALUE
000464  027          AGAIN:    RAL               ;ROTATE INTO CARRY
000465  043                    INX    H          ;BUMP TABLE POINTER
000466  322 054 001            JNC    AGAIN      ;
000471  116                    MOV    C,M        ;SAVE KEY CODE

000472  072 305 007            LDA    OFLAG      ;CHECK HEX/OCT FLAG
000475  267                    ORA    A          ;SET FLAGS
000476  302 102 001            JNZ    OCT2       ;IF IN OCTAL MODE JUMP TO CHAR CHECK
000501  311                    RET

000502  171          OCT2:     MOV    A,C        ;GET KEY VALUE
000503  346 370                ANI    370Q         ;MASK OFF LOWER DIGIT
000505  310                    RZ                ;RETURN IF LEGAL OCTAL NUMBER
000506  303 131 000            JMP    KEY        ;ILLEGEL CHAR GOTO  KEY

                     ;********ROUTINE TO READ KEYBOARD******

000511  072 007 005  READ:     LDA    KEYBD      ;READ KEYBOARD
000514  057                    CMA               ;COMPLEMENT
000515  267                    ORA    A          ;SET FLAGS
000516  311                    RET

                     ;*******ROUTINE TO DISPLAY HEX OR OCTAL********

000517  072 267 007  DIS:      LDA    KYTEM      ;GET CURRENT DISPLAY VALUE
000522  117          DISPLAY:  MOV    C,A        ;SAVE A REG
000523  072 305 007            LDA    OFLAG      ;CHECK HEX/OCT FLAG
000526  267                    ORA    A          ;SET FLAGS
000527  302 136 001            JNZ    OCT        ;SIGN BIT=1 FOR OCT DISPLAY
000532  171          HEX:      MOV    A,C        ;HEX - GET AC
000533  323 000                OUT    0          ;SEND TO DISPLAY
000535  311                    RET
000536  171          OCT:      MOV    A,C        ;GET NUMBER TO DISPLAY
000537  007                    RLC               ;GET HIGH ORDER DIGIT
000540  007                    RLC               ;ROTATE INTO POSITION
000541  346 003                ANI    3Q         ;SAVE HIGH ORDER DIGIT
000543  323 005                OUT    5          ;DISPLAY HIGH ORDER DIGIT
000545  171                    MOV    A,C        ;GET NUMBER AGAIN
000546  027                    RAL               ;MOVE 2ND DIGIT INTO POSTION
000547  346 160                ANI    160Q       ;SAVE MIDDLE DIGIT
000551  107                    MOV    B,A        ;SAVE MIDDLE DIGIT
000552  171                    MOV    A,C        ;GET NUMBER AGAIN
000553  346 007                ANI    7Q         ;GET 1ST DIGIT
000555  260                    ORA    B          ;COMBINE DIGITS 1 & 2
000556  323 000                OUT    0          ;DISPLAY THEM
```

```
000560   311                        RET

                                    ;*******THIS IS A DELAY ROUTINE TO DEBOUNCE THE SWITCHES*****

000561   006 000         DELAY:     MVI     B,0               ;INITIALIZE COUNTER
000563   004             LOOP:      INR     B                 ;BUMP COUNTER
000564   343                        XTHL                      ;EXTRA DELAY IN LOOP
000565   343                        XTHL
000566   302 163 001                JNZ     LOOP              ; LOOP UNTIL ZERO
000571   311                        RET

000572   000             TABLE:     DB      00H               ;NUMBER KEY CODE TABLE
000573   004                        DB      04H
000574   010                        DB      08H
000575   014                        DB      0CH
000576   001                        DB      01H
000577   005                        DB      05H
000600   011                        DB      09H
000601   015                        DB      0DH
                                    END
NO PROGRAM ERRORS
```



Flow chart for
HEX/OCT monitor program for MST-80B
Microcomputer trainer.