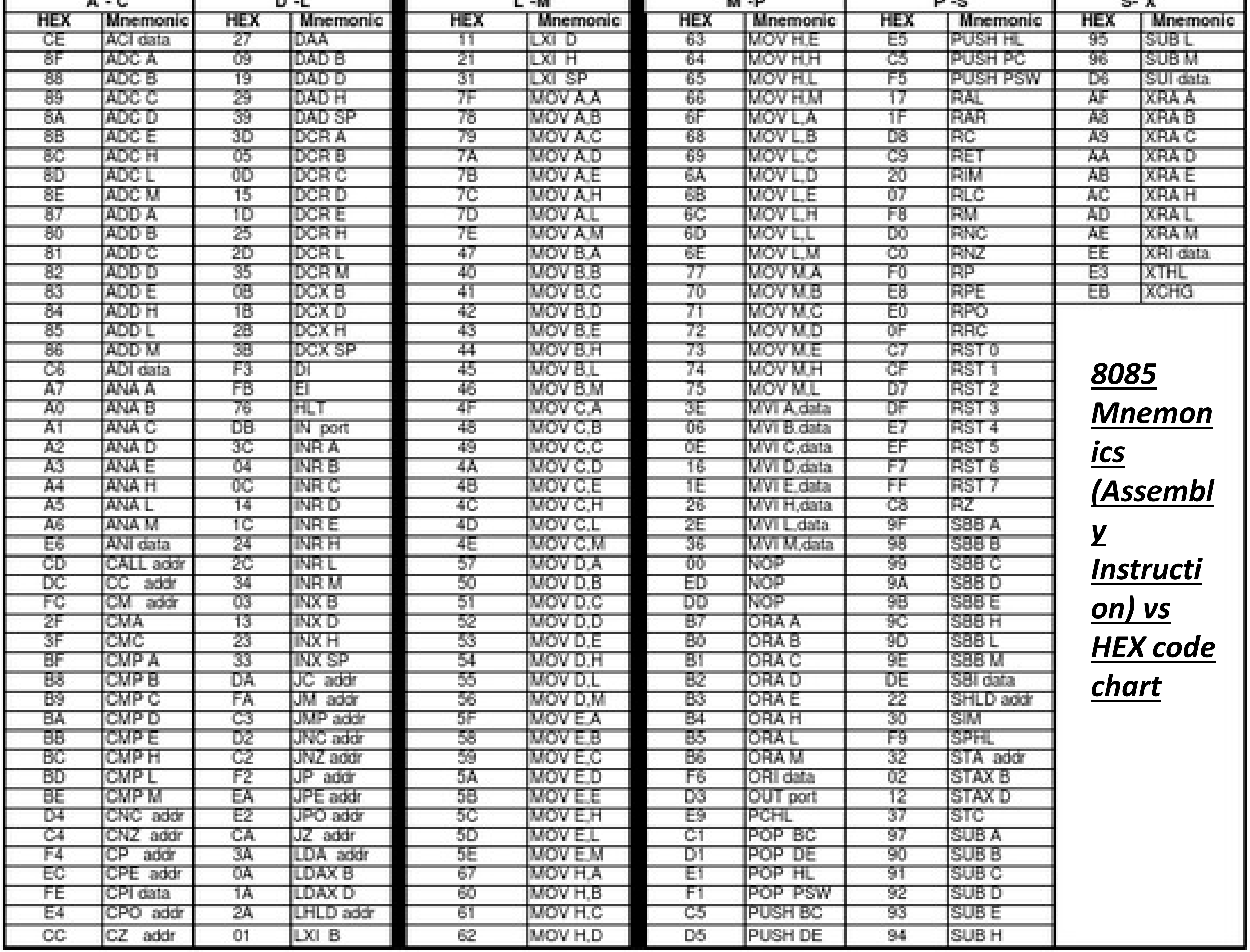
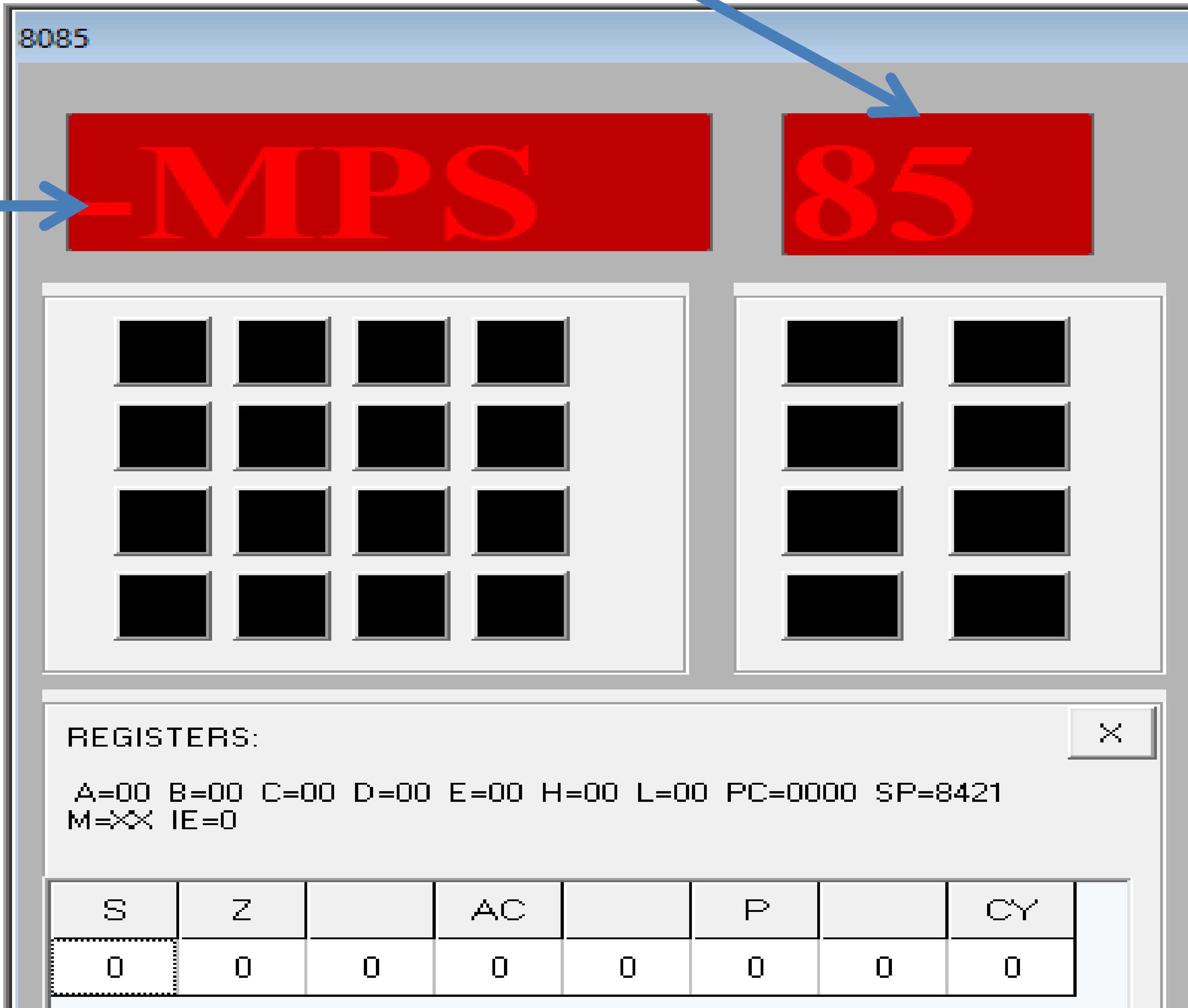


A - C		D - L		L - M		M - P		P - S		S - X	
HEX	Mnemonic	HEX	Mnemonic	HEX	Mnemonic	HEX	Mnemonic	HEX	Mnemonic	HEX	Mnemonic
CE	ACI data	27	DAA	11	LXI D	63	MOV H,E	E5	PUSH HL	95	SUB L
8F	ADC A	09	DAD B	21	LXI H	64	MOV H,H	C5	PUSH PC	96	SUB M
88	ADC B	19	DAD D	31	LXI SP	65	MOV H,L	F5	PUSH PSW	D6	SUI data
89	ADC C	29	DAD H	7F	MOV A,A	66	MOV H,M	17	RAL	AF	XRA A
8A	ADC D	39	DAD SP	78	MOV A,B	6F	MOV L,A	1F	RAR	A8	XRA B
8B	ADC E	3D	DCR A	79	MOV A,C	68	MOV L,B	D8	RC	A9	XRA C
8C	ADC H	05	DCR B	7A	MOV A,D	69	MOV L,C	C9	RET	AA	XRA D
8D	ADC L	0D	DCR C	7B	MOV A,E	6A	MOV L,D	20	RIM	AB	XRA E
8E	ADC M	15	DCR D	7C	MOV A,H	6B	MOV L,E	07	RLC	AC	XRA H
87	ADD A	1D	DCR E	7D	MOV A,L	6C	MOV L,H	F8	RM	AD	XRA L
80	ADD B	25	DCR H	7E	MOV A,M	6D	MOV L,L	D0	RNC	AE	XRA M
81	ADD C	2D	DCR L	47	MOV B,A	6E	MOV L,M	C0	RNZ	EE	XRI data
82	ADD D	35	DCR M	40	MOV B,B	77	MOV M,A	F0	RP	E3	XTHL
83	ADD E	0B	DCX B	41	MOV B,C	70	MOV M,B	E8	RPE	EB	XCHG
84	ADD H	1B	DCX D	42	MOV B,D	71	MOV M,C	E0	RPO		
85	ADD L	2B	DCX H	43	MOV B,E	72	MOV M,D	0F	RRC		
86	ADD M	3B	DCX SP	44	MOV B,H	73	MOV M,E	C7	RST 0		
C6	ADI data	F3	DI	45	MOV B,L	74	MOV M,H	CF	RST 1		
A7	ANA A	FB	EI	46	MOV B,M	75	MOV M,L	D7	RST 2		
A0	ANA B	76	HLT	4F	MOV C,A	3E	MVI A,data	DF	RST 3		
A1	ANA C	DB	IN port	48	MOV C,B	06	MVI B,data	E7	RST 4		
A2	ANA D	3C	INR A	49	MOV C,C	0E	MVI C,data	EF	RST 5		
A3	ANA E	04	INR B	4A	MOV C,D	16	MVI D,data	F7	RST 6		
A4	ANA H	0C	INR C	4B	MOV C,E	1E	MVI E,data	FF	RST 7		
A5	ANA L	14	INR D	4C	MOV C,H	26	MVI H,data	C8	RZ		
A6	ANA M	1C	INR E	4D	MOV C,L	2E	MVI L,data	9F	SBB A		
E6	ANI data	24	INR H	4E	MOV C,M	36	MVI M,data	98	SBB B		
CD	CALL addr	2C	INR L	57	MOV D,A	00	NOP	99	SBB C		
DC	CC addr	34	INR M	50	MOV D,B	ED	NOP	9A	SBB D		
FC	CM addr	03	INX B	51	MOV D,C	DD	NOP	9B	SBB E		
2F	CMA	13	INX D	52	MOV D,D	B7	ORA A	9C	SBB H		
3F	CMC	23	INX H	53	MOV D,E	B0	ORA B	9D	SBB L		
BF	CMP A	33	INX SP	54	MOV D,H	B1	ORA C	9E	SBB M		
B8	CMP B	DA	JC addr	55	MOV D,L	B2	ORA D	DE	SBI data		
B9	CMP C	FA	JM addr	56	MOV D,M	B3	ORA E	22	SHLD addr		
BA	CMP D	C3	JMP addr	5F	MOV E,A	B4	ORA H	30	SIM		
BB	CMP E	D2	JNC addr	58	MOV E,B	B5	ORA L	F9	SPHL		
BC	CMP H	C2	JNZ addr	59	MOV E,C	B6	ORA M	32	STA addr		
BD	CMP L	F2	JP addr	5A	MOV E,D	F6	ORI data	02	STAX B		
BE	CMP M	EA	JPE addr	5B	MOV E,E	D3	OUT port	12	STAX D		
D4	CNC addr	E2	JPO addr	5C	MOV E,H	E9	PCHL	37	STC		
C4	CNZ addr	CA	JZ addr	5D	MOV E,L	C1	POP BC	97	SUB A		
F4	CP addr	3A	LDA addr	5E	MOV E,M	D1	POP DE	90	SUB B		
EC	CPE addr	0A	LDAX B	67	MOV H,A	E1	POP HL	91	SUB C		
FE	CPI data	1A	LDAX D	60	MOV H,B	F1	POP PSW	92	SUB D		
E4	CPO addr	2A	LHLD addr	61	MOV H,C	C5	PUSH BC	93	SUB E		
CC	CZ addr	01	LXI B	62	MOV H,D	D5	PUSH DE	94	SUB H		

<u>8085 Mnemonics (Assembly Instruction) vs HEX code chart</u>

Data field of Display , Generally displays *content of the Memory location* (displayed on left hand side), in 2 Hex digits (8 binary bits) (8085's Data bus width is of 8 bit)

Address field of Display , Generally displays Address of a Memory location in 4 Hex digits (16 bi binary bits) (8085's Address bus width is of 16 bit) [8085 can support 64K 2^{16} addressable memory locations]



REGISTERS:

A=00 B=00 C=00 D=00 E=00 H=00 L=00 PC=0000 SP=8421
M=XX IE=0

S	Z		AC		P		CY
0	0	0	0	0	0	0	0

Layout of Push button Switch of 8085 simulator

C	D	E	F	Reset	KBint
8	9	A	B	Previous	Exam Memory
4	5	6	7	Next	Exam Register
0	1	2	3	Go	Exec

Register Structure of 8085

- Accumulator **A (ACC)**, is an special as well general purpose8-bit register. It is the 1st operand & result register for most of arithmatic & logical operations.
- An 8 bit **Flag register** contain 5 Flag Flip Flop status, which always remains updated by the result of latest Arithmatic or Logical operation result, used by branch control instructions.
- Register **B**, **C**, **D**, **E**, **H**, and **L**, are six 8-bit general purpose register. These registers can be accessed individually, or can be accessed in pairs **BC**, **DE**, **HL,** by some of the instructions.
- Register **HL** is used to point to a memory location.
- Stack pointer, **SP**, is an 16-bit register, which contains the address of the top of the stack.
- *Program Status Word (PSW):* Accumulator and Flag Register can be combined as a register pair called PSW

Flag FFs (Flip Flops)(5) reflect data conditions of latest Arithmatic 0r Logical operation & are stored in 5 bit positions of an 8 bit Register

- The ***sign flag, S*** =1(may indicate result is -ve), if D7 bit of result (of an arithmatic operation) =1. indicates the sign of a value of result(as in 2s complement arithmatic, MSB indicates Siign bit). If signed no.s are not used, this bit don't have any significance.
- The ***zero flag, Z***, is set to 1 if an arithmetic or logical operation produces a result of 0; otherwise set to 0.
- ***parity flag, P***, is set to 1 if the result of an arithmetic or logical operation has an even number of 1's; otherwise it is set to 0.
- The ***carry flag, CY***, is set when an arithmetic operation generates a carry out.
- The ***auxiliary carry flag, AC***, very similar to CY, but it denotes a carry from the lower half of the result to the upper half. (if acarry generated from D3 bit position of result)

S	**Z**	**X**	**AC**	**X**	**P**	**X**	**CY**

Flag Register

- 8085 up has another 8 bit special purpose register “Instruction Register” (IR). Not accessible by user or programmer. After fetching the opcode (first byte of a machine instruction), from memory, it is stored in IR, for subsequent decoding of opcode by Instruction decoder

Accumulator (8 bit)	**Flags (8 bit)**
B (8 bit)	**C (8 bit)**
D (8 bit)	**E (8 bit)**
H (8 bit)	**L (8 bit)**
Program Counter (16 bit)	
Stack Pointer (16 bit)	

Data movement instruction for the 8085 microprocessor

Instruction	**Operation**
MOV r1, r2	r1 = r2 [r2 register content is copied into r1 register]
LDA Γ Memory read operation by 8085	A = M[Γ] [Load ACC with the content of memory location (whose 16 bit address is Γ)]
STA Γ Memory read operation by 8085	M[Γ] = A [Store content of A(ACC) reg. into a memory location (whose 16 bit address is Γ)]
MVI Rd, 8~bit	Load 8-bit immediate data in a register.
IN n	A = input devices data whose address is n
OUT n	Output devices data whose address is n =A

r, r1, r2 – any 8-bits register

Γ–16 bit memory locations address

M[Γ] – Content of a memory location, whose 16 bit address is Γ

n – 8-bit Input/ Output device address

Data operation instruction for the 8085 microprocessor

CY – carry flag

Instruction	Operation	Flags
ADD r	A = A + r	All
ADD M	A = A + M[HL]	All
INR r	r = r + 1 *[increment an 8 bit register ontent]* *[A reg. (Accumulator) not modified]*	Not CY
INR M	M[HL] = M[HL] + 1 *[ACC not modified]*	Not CY
DCR n	r = r – 1 *[decrement an 8 bit register content]* *[ACC not modified]*	Not CY
DCR M	M[HL] = M[HL] – 1 *[ACC not modified]*	Not CY
XRA r	A = A ⊕ r *[⊕ stands for XOR]*	All
XRA M	A = A ⊕ M[HL]	All
SUB r	A = A - r	All
CMA	A = A′	None

Program1: Let us write a program to do the following operations

1) Let an 8 bit data "76H" be immediately loaded into B register
2) Let B Register content be copied (Moved) into A register
3) Let A register content be copied into a Memory location 8050H

Hence our program consists of following Instructions

MVI D, 76H	16H, 76H	2 Byte Instruction
MOV A, D	7AH	2 Byte Instruction
STA 8050H	32H, 8050H	3 Byte Instruction

- [H stands for Hexadecimal notation][opcode of MVI D is 16H] [opcode of MOV A, D is 7AH] [opcode of STA is 32H]- as obtained from Hex chart]
- [Clearly MVI D, 76H is a 2 byte instruction, as the opcode of MVI D should be followed by an 8 bit data (76H) & one memory location can contain only 8 bit data][opcode is the first byte of an instruction]
- [Clearly STA 8050H is a 3 byte instruction, as the opcode of STA should be followed by a 16 bit (2 byte) memory location address]
- [Clearly MOV A, D is a 1 byte instruction, as the opcode of MOV A, D need not to be followed by an additional 8 bit data or 16 bit memory address or data.] [Here the opcode of MOV A, D itself specifies operation code, destination & source operand address.]

MVI D, 76H	16H, 76H
MOV A, D	7AH
STA 8050H	32H, 8050H
HLT	76H

To read a 16t bit data from memory, 8085 at first read low order byte from memory, store it in Z reg., then High high order byte (in W reg.), so program entered that way .

[HLT instruction represent end of Program, opcode of HLT is 76H (from 8085 Hex chart)]

Let the starting address of our program be 8000H. Hence we store our program in memory in the following way:

Memory location	Memory-content Machine Instruction (binary/ Hex)	Assembly Language Instruction	Comment
8000H	16H	MVI D, 76H	//Move 76H into D reg.
8001H	76H		
8002H	7AH	MOV A, D	// A = D
8003H	32H	STA 8050H	// M[8050H] = A
8004H	50H		
8005H	80H		
8006H	76H	HLT	// End of Program

In 8085 Simulator, or in 8085 trainer kit we have in our Lab:

(A)For entering Program, in Memory with Program starting address say 8000H, do following steps:

(1).RESET (2). Exam Memory (3). type starting address of your program (8000) using hex keyboard (or by Laptop Keyboard)
[see address field displays 8000]
(4). Next (5). type opcode of first instruction (16)
[see it will be displayed in data field]
(6). Next [see address field displays next memory location address]
(7). Type second machine code byte (76)
[see it (76) will be displayed in data field of display, see it is being displayed as 8001 memory locations content]
& So on Enter rest of the program.

[IMPortant: Don't forget to press "NEXT" after entering the last Instruction HLT (76), other last entry will not be finally upodated in corresponding memory location]

(B)For Executing a Program(after entering program)do following steps:

1. RESET **2. GO**
3. type starting address of your program (8000) **4. Exec**

After Executing the program, see if the address field displays E, indicates Program successfully executed.

(C) If you want to check any memory locations content, do following steps:

(1)Exam Memory (2). type four hex digit (16 binary bit) Memory address (3). Next

After executing the program check, whether 8050 Memory location content is 76